

8/29/95
E-9829

NASA Contractor Report 198375

SSME Post Test Diagnostic System Systems Section

Timothy Bickmore
Aerojet Propulsion Systems
Sacramento, California

August 1995

Prepared for
Lewis Research Center
Under Contract NAS3-25883



National Aeronautics and
Space Administration



Intelligent Software Associates, Inc.

SSME Post Test Diagnostic System Systems Section

Final Report

**Task 11 of Contract NAS3-25883
*Development of Life Prediction Capabilities for
Liquid Propulsion Rocket Engines***

Prepared for:

**Aerojet Propulsion Systems
P.O. Box 13222
Sacramento, CA 95813-6000**

Submitted by:

**Intelligent Software Associates, Inc.
P.O. Box 188825
Sacramento, CA 95818**

May 6th, 1995

Contents

I. Introduction.....	1
I.1. The SSME Post-Test Diagnostic System Project.....	1
I.2. The SSME PTDS Systems Section.....	2
I.3. Results.....	5
I.4. Other PTDS Enhancements.....	6
I.4.1 Further Enhancements to the HPOTP Module.....	6
I.4.2. Porting the Anomaly Database from Ingres to TekBase.....	7
I.4.3. Integration of "Features" Signal Processing Routines.....	7
I.4.4. TKCLIPS Training	7
II. Systems Section Architecture.....	8
II.1. Feature Extractor Module.....	8
II.2. Sensor Validation Module	11
II.3. Hardware Change Module.....	11
II.4. External Effects Module	12
II.5. Case-Based Reasoner Module	12
II.5. Performance Module	14
III. Anomalies Currently Detected by the Systems Section.....	15
IV. Conclusion.....	19
IV.1. Future Work.....	19
Acknowledgements.....	21
References	22
 User's Guide	 Attachment #1
Programmer's Guide	Attachment #2
Transcripts of Interviews with SSME Data Analysts.....	Attachment #3
GENERIC Features Description and TKCLIPS User's Guide	Attachment #4

I. Introduction

An assessment of engine and component health is routinely made after each test firing or flight firing of a Space Shuttle Main Engine (SSME). Currently, this health assessment is done by teams of engineers who manually review sensor data, performance data, and engine and component operating histories. Based on review of information from these various sources, an evaluation is made as to the health of each component of the SSME and the preparedness of the engine for another test or flight.

The objective of this project—the SSME Post-Test Diagnostic System (PTDS)¹—is to develop a computer program which automates the analysis of test data from the SSME in order to detect and diagnose anomalies. This report primarily covers work on the Systems Section of the PTDS, which automates the analyses performed by the systems/performance group at the Propulsion Branch of NASA Marshall Space Flight Center (MSFC). This group is responsible for assessing the overall health and performance of the engine, and detecting and diagnosing anomalies which involve multiple components (other groups are responsible for analyzing the behavior of specific components).

The PTDS utilizes several advanced software technologies to perform its analyses. Raw test data is analyzed using signal processing routines which detect features in the data, such as spikes, shifts, peaks, and drifts. Component analyses are performed by expert systems, which use “rules-of-thumb” obtained from interviews with the MSFC data analysts to detect and diagnose anomalies. The systems analysis is performed using case-based reasoning. Results of all analyses are stored in a relational database and displayed via an X-window-based graphical user interface which provides ranked lists of anomalies and observations by engine component, along with supporting data plots for each.

I.1. The SSME Post-Test Diagnostic System Project

The post-test diagnostic system is a cooperative effort involving engineers and scientists at NASA Marshall Space Flight Center (MSFC), NASA Lewis Research Center (LeRC), Aerojet Propulsion Systems, Intelligent Software Associates, Inc. (ISAI), Science Applications International Corporation (SAIC), and support contractors from NYMA, Analex, Computer Sciences Corp., and Martin Marietta. Work on the PTDS began in 1990, and development of the Systems Section was started in 1992.

The PTDS is designed to automate post-test firing data reviews. The first application has been to the space shuttle main engine. A modular, distributed architecture was selected which enables the use of separate modules which analyze different aspects of an engine's performance. The PTDS modules currently implemented or being developed include the following (see Figure 1):

- CAE Package — The Computer Aided Engineering package is used primarily to provide a very flexible mechanism for displaying plots of engine data. The PV-Wave command language was selected as a commercial off-the-shelf (COTS) package to fill this need.
- Relational Database Management System — A database is used to store information about tests, engines configurations, anomalies, performance parameter histories, and all PTDS analysis results (using the TekBase relational database system).
- Session Manager — The executive for the system which launches each of the modules as needed once test data becomes available. Implemented in C.
- Feature Extractor — Performs the pattern recognition analyses on the raw data. Implemented in C (routines can also be called from within CLIPS).

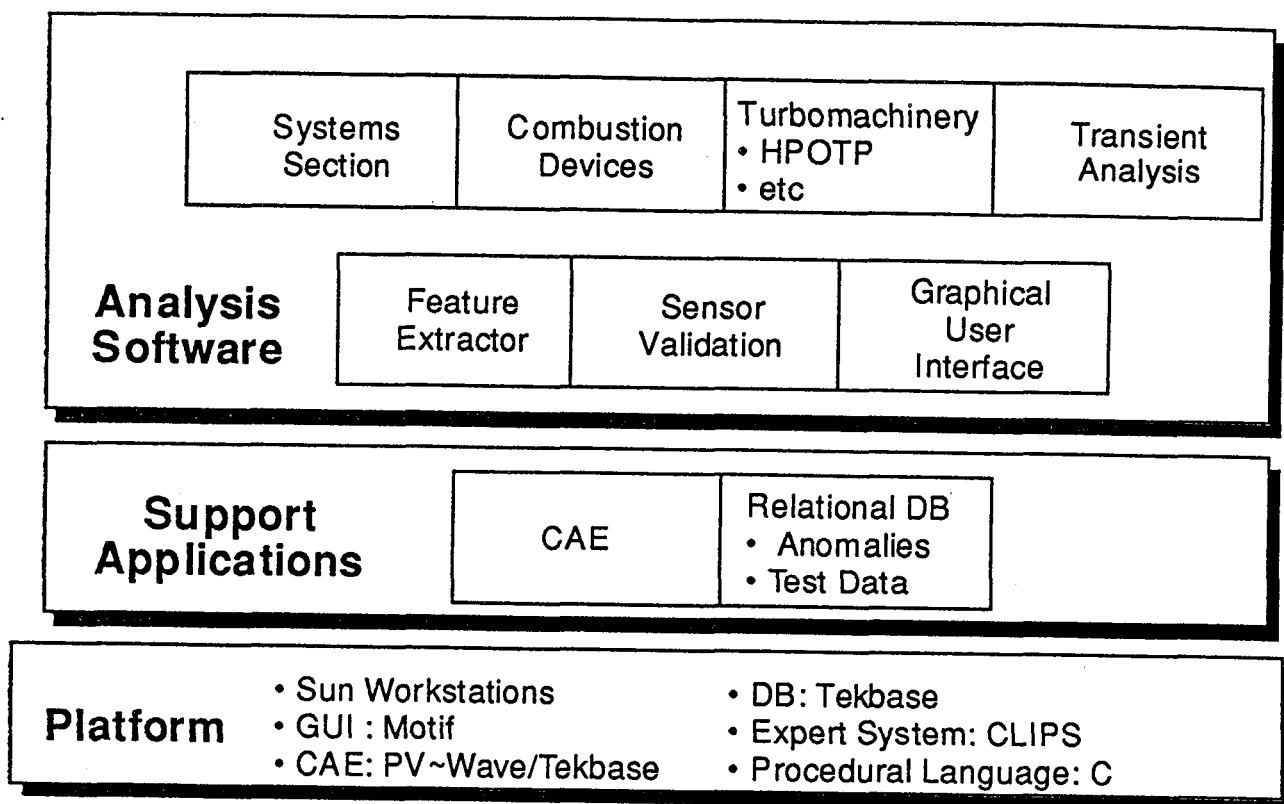


Figure 1. PTDS Architecture

- Sensor Validation — Detects instrumentation anomalies and failures so that analysis modules do not reason over bad data.
- Systems Section — Analyzes the system-wide health and performance of the engine, and detects anomalies which involve more than one component. Implemented in C and CLIPS.
- HPOTP Analysis Module — Analyzes the health and performance of the SSME HPOTP. Implemented in CLIPS.
- Combustion Devices Module — Detects anomalies in the main combustion chamber and the two pre-burners. Implemented in CLIPS and scheduled for completion in 1995.
- Transient Module — Detects anomalies during start and shutdown transients. Implemented in CLIPS and scheduled for completion in 1995.
- HPFTP, LPFTP, LPOTP Analysis Modules — Analyzes the health and performance of the other three turbopumps on the SSME. To be implemented in CLIPS and scheduled for completion in 1996.
- Graphical Browser — Allows analysts to view PTDS results using an X-windows-based display (see Figure 2). Implemented in C.

I.2. The SSME PTDS Systems Section

Development of the Systems Section was begun in 1992 with a series of lengthy interviews with data analysts from MSFC's propulsion branch systems/performance group. The interviews were initially conducted via telephone, but a series of on-site visits were made in

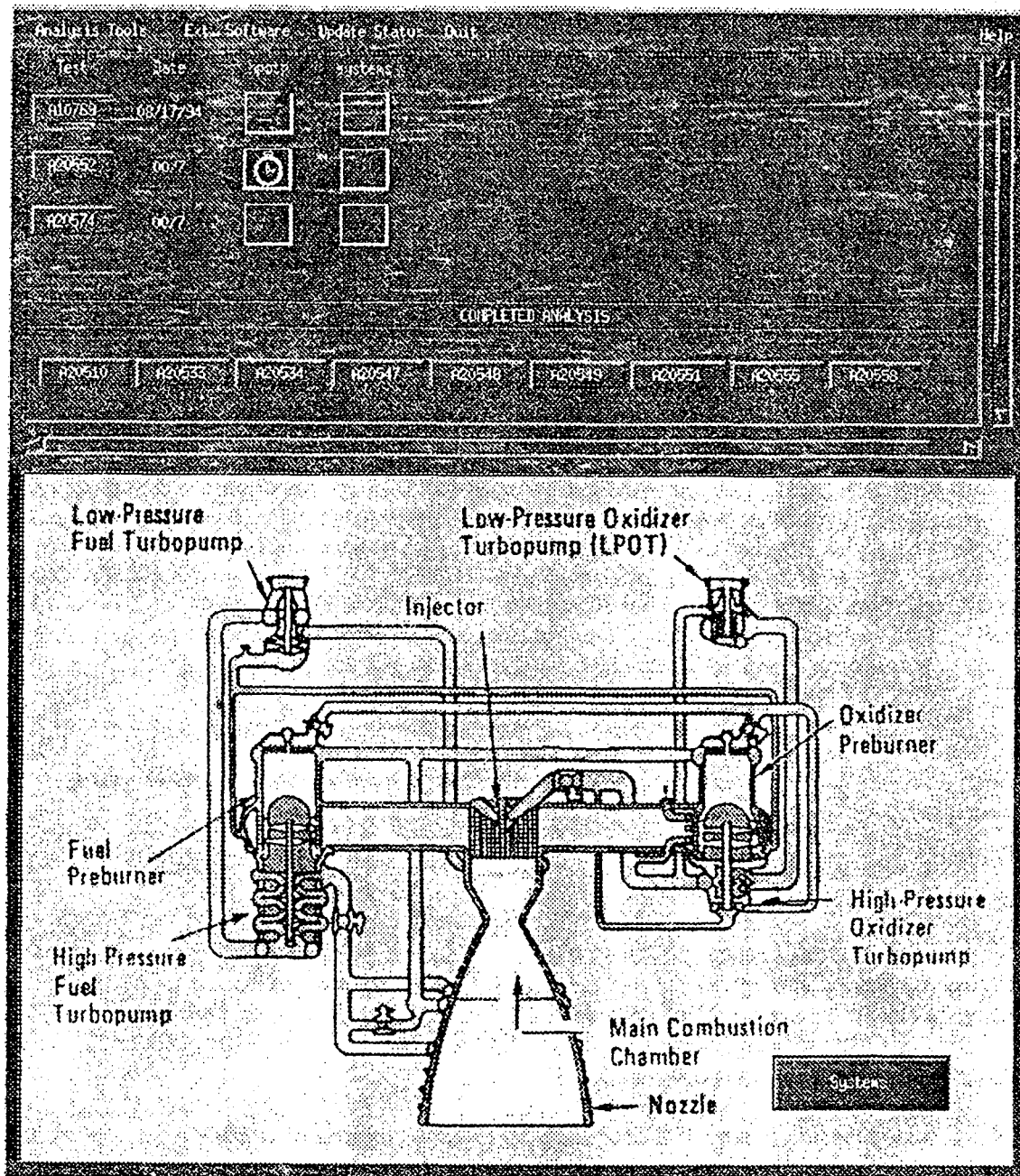


Figure 2. PTDS Graphical User Interface

April and May of 1992 during which the analysts were asked to walk-through their analyses of earlier tests and describe the process and reasoning they used (transcripts from these interviews are given in Attachment #3).

The strategy taken by the systems analysts was to first detect an anomaly, then to try and explain it. Anomaly detection primarily consisted of comparing the current test to one or more previous tests using plots with overlaid data (see Figure 3) and noting any significant deviations in the current test data. To explain any detected discrepancies, analysts would generally go through the following steps:

1. Determine if the anomaly is "real" — This is the problem of sensor validation. The first thing analysts will typically do upon detection of an event on a plot is to cross-check it with the plots of related engine parameters. If there is at least one other signal which confirms that something changed at the point of the event, then the event is considered real, otherwise a failed sensor is suspected.
2. Determine if the discrepancy is due to differences in hardware configuration between the comparison tests and the current test — Each engine is unique, and has its own performance characteristics (e.g., different pump efficiencies, line resistances, leak rates, etc.), and since the SSME is built-up with Line-Replaceable Units (LRUs) which are very frequently changed, even consecutive tests of the same engine can show substantial differences in performance. In comparing a current test to a comparison test these differences must be taken into account in order to "explain away" discrepancies which are due to such hardware changes.
3. Determine if the anomaly is due to some influence which is external to the engine — Differences in thrust profiles, vent profiles, mixture ratios, or repressurization flows will all lead to significant differences in sensor data between two tests. These "external effects" must be taken into account when determining if an anomaly represents a problem or simply a change in operating conditions.
4. If none of the above steps provides a valid explanation for a discrepancy, then the discrepancy is labelled an anomaly, and a diagnostic investigation is made into its possible causes. Most of the diagnostic reasoning used by the analysts utilize what they call "gains models", which are monotonic qualitative causal models of the form "parameter X varies directly with Y and inversely with Z" (e.g., "HPFTP speed increases with volumetric flow, decreases with efficiency, decreases with downstream resistance, and decreases with pump inlet pressure."). These models are used to isolate a set of likely candidates which could have caused the anomaly.

This overall diagnostic strategy was taken as the blueprint for the design of the Systems Section software.² The process starts with the Sensor Validation module which attempts to detect failed or anomalous sensors and remove them from further consideration by the system. Next, the current test data is compared with data from a previously conducted test, which has been adjusted for differences in hardware configuration and external effects. A signal processing routine is then run on each of the difference signals to detect significant shifts in behavior between the two tests. Finally, a "Case-Based Reasoner" is employed to determine a set of candidate causes for the observed shifts, using quantitative gains values derived from runs of the SSME Power Balance Model and heuristic information supplied by the expert data analysts. This set of candidate failure causes is then written to the database along with descriptions of supporting plots for later viewing by the analysts. Note that analyses are performed during steady-state operation of the engine.

One element of the Systems Section does not follow the above strategy. The accurate determination of pump and turbine efficiencies requires extensive calculations, so a separate Performance Module was implemented specifically to detect subtle shifts in these

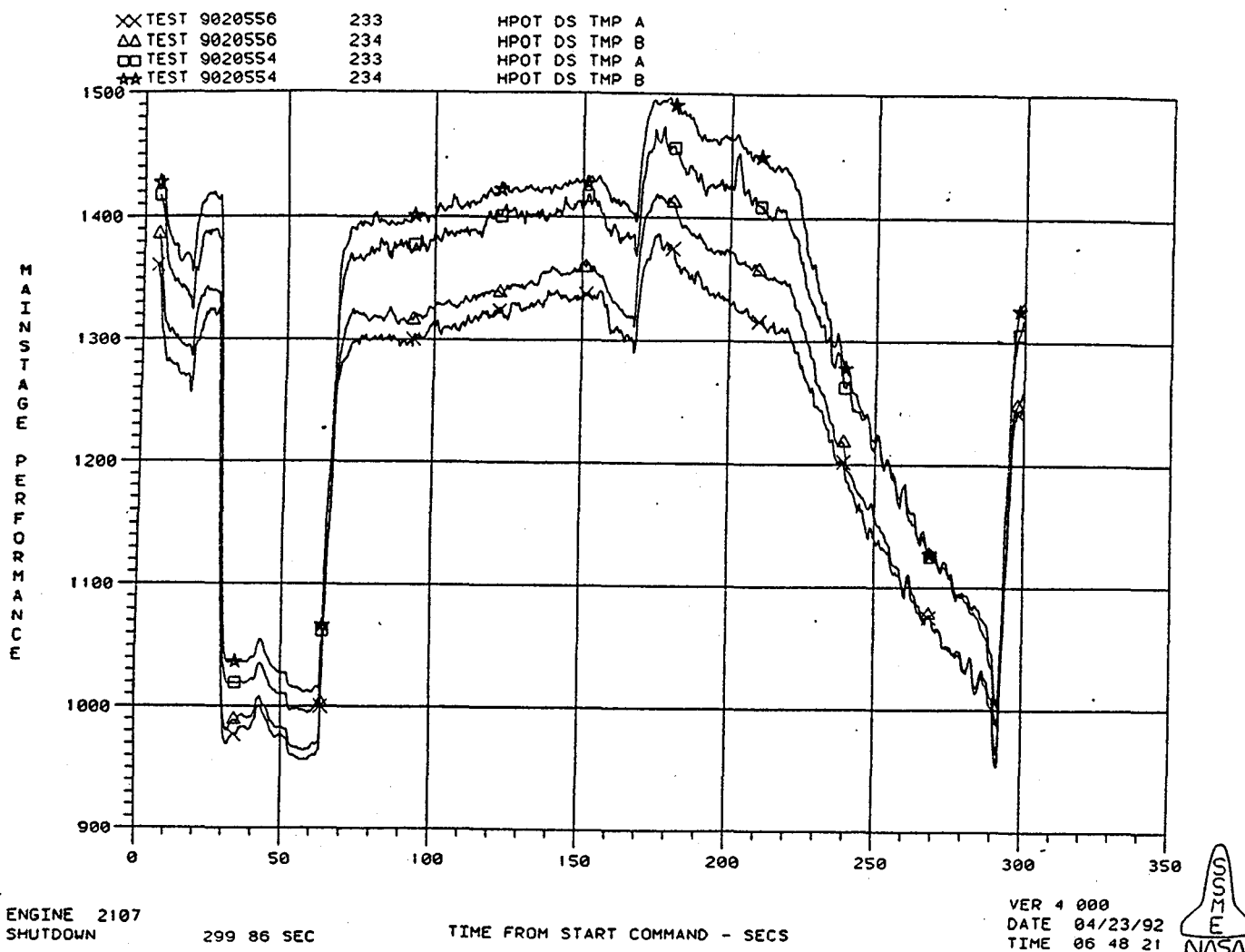


Figure 3. Example Current-vs-Comparison Test Plot

performance characteristics. The Performance Module consists of a large portion of the SSME Power Balance Model which computes performance characteristics at one-second intervals during steady-state, and detects any significant shifts in these parameters over time.

I.3. Results

The Systems Section was demonstrated to several systems analysts at MSFC on February 27th and 28th, 1995. During this demonstration the system correctly detected and diagnosed a Piston Ring Seal Shift anomaly on SSME test A40019. During these demonstrations the analysts provided feedback which will be used to guide future refinements of the system.

I.4. Other PTDS Enhancements

Several additional tasks were performed under this contractual effort which were not directly part of the Systems Section.

I.4.1 Further Enhancements to the HPOTP Module

Several additional enhancements were made to the HPOTP module,³ including:

- Extended analyses to cover the Pratt & Whitney Advanced Technology Development (ATD) pumps. This involved the creation of a new database table to track performance statistics for ATD pumps, classification of the ATD into hot or cold "ski slope" categories, modification of 44 diagnostic rules, and the addition of 17 greenrun specification checks.
- Improved the accuracy of detecting nose seal leaks by implementing a new signal processing routine in the feature extractor to identify them. These features are extremely subtle and difficult to accurately detect, since they involve shifts whose magnitude is on the order of the sensor's noise (i.e., two or three "bit-toggles").
- Added a sensor validation check for redundant sensors which drift apart during steady-state.
- Fixed the greenrun specification check for duration of minimum and maximum LOX inlet NPSP conditions by extending the analysis across power level transients.
- Extended the HPOTP module so that it would run from a Unix command-line so that it could be added to MSFC's "run-stream" and executed automatically following each test.
- Added the capability for the HPOTP module to generate a textual report similar to the one-page test summaries produced by the turbomachinery group (Figure 4. shows an example).

```
SSME Post-Test Diagnostic System -- HPOTP Analysis
Test: A10750 Duration: 710 HPOTP Type: P&W
Analysis Run 03/10/95 by rballard

Instrumentation
-----
328, 2, 518, 521, 519, 522, 327 -- Do not exist in datafile.
1188 -- Not at post-test ambient conditions.
211/212 -- Drift apart between 18.00 and 94.00.

GreenRun Specification Check (Note: This is NOT a green run.)
-----
Failed HPOTP GreenRun 65/64/63% throttle criteria 3.5.1.2(d)

Anomalies
-----
Seen between T=21.00 and 35.00. Possible rotor drag.

Observations
-----
HOT ski-slope classification.
```

Figure 4. Sample HPOTP Text Report

I.4.2. Porting the Anomaly Database from Ingres to TekBase

The Anomaly Database is a graphical user interface to a set of database tables which allow data analysts to store information about observed anomalies, and to search the database using a variety of selection criteria. With assistance from personnel at NASA LeRC, this database was ported from Ingres to TekBase (the relational database being used by the MSFC SSME data analysts).

I.4.3. Integration of "Features" Signal Processing Routines

During the development of the Enhanced HPOTP Diagnostic Module, the signal processing routines in the Features Module were copied and integrated into CLIPS so that the HPOTP module could be run as a stand-alone system. In the year that followed, many changes were made to both sets of routines resulting in a significant software maintenance problem. The solution to this problem was to create a single source file of "generic" signal processing routines which could be called either from CLIPS or from the Features Module. This involved the integration of some 11,000 lines of C code. The result, described in Attachment #4, is a set of routines which can be centrally maintained. In addition, these routines can easily be integrated into any future health monitoring application which needs to analyze time-varying data.

I.4.4. TKCLIPS Training

A training seminar was held at NASA MSFC, March 1st - 3rd, 1995, for NASA personnel and support contractors working on the PTDS project.

II. Systems Section Architecture

This section describes the overall architecture of the Systems Section. Figure 5 shows a high-level dataflow diagram for the PTDS, and Figure 6. shows the current execution sequence of the modules in the system. Each of the modules in the Systems Section is described next.

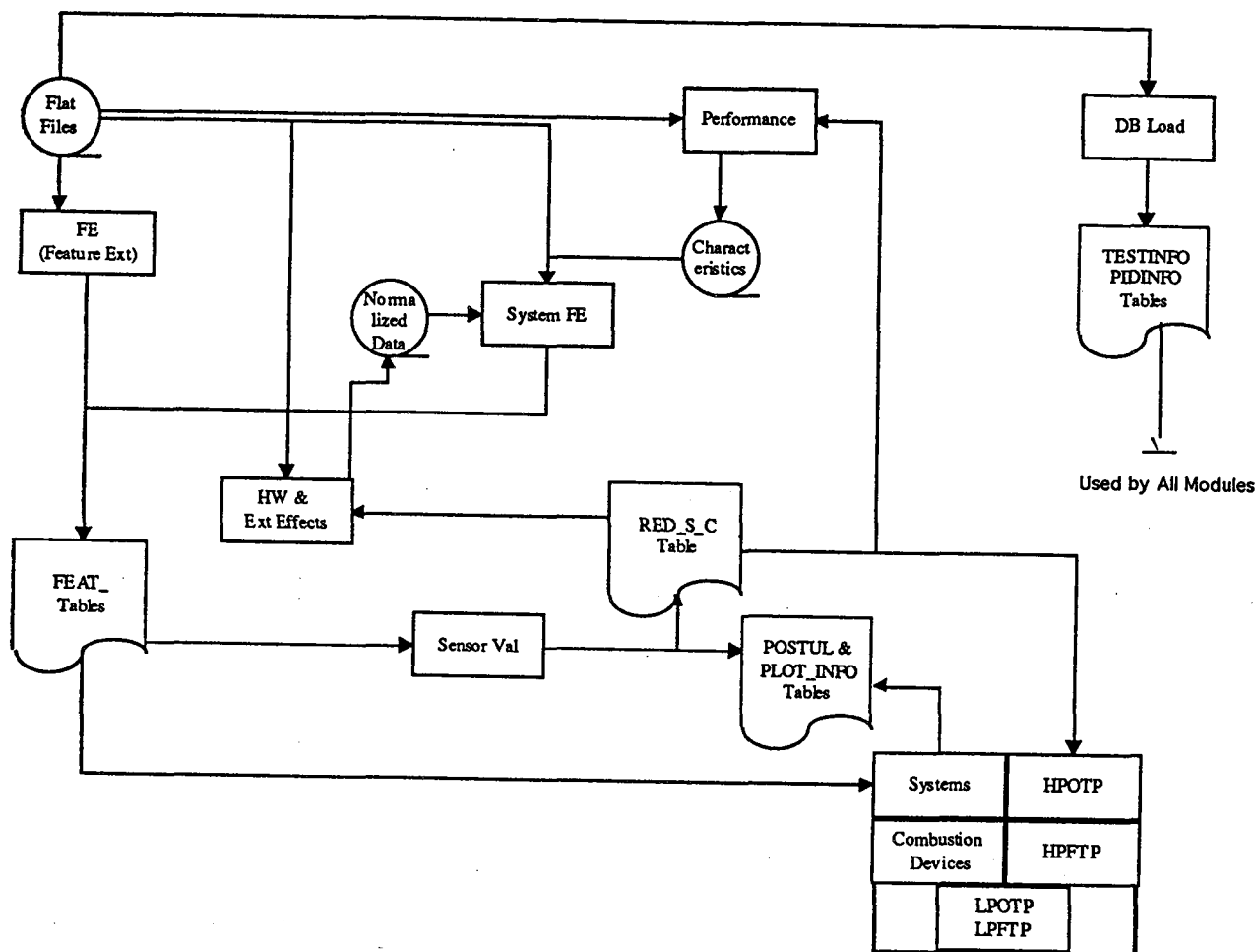


Figure 5. PTDS Dataflow

II.1. Feature Extractor Module

The Feature Extractor employs signal processing routines which analyze raw time-varying data from an SSME test firing to detect features of interest for the rest of the PTDS. The features currently detected by this module are shown in Table 1. The Feature Extractor imports descriptions of the features to look for from database tables, and writes any features found back out to the database for use by the other modules in the PTDS.

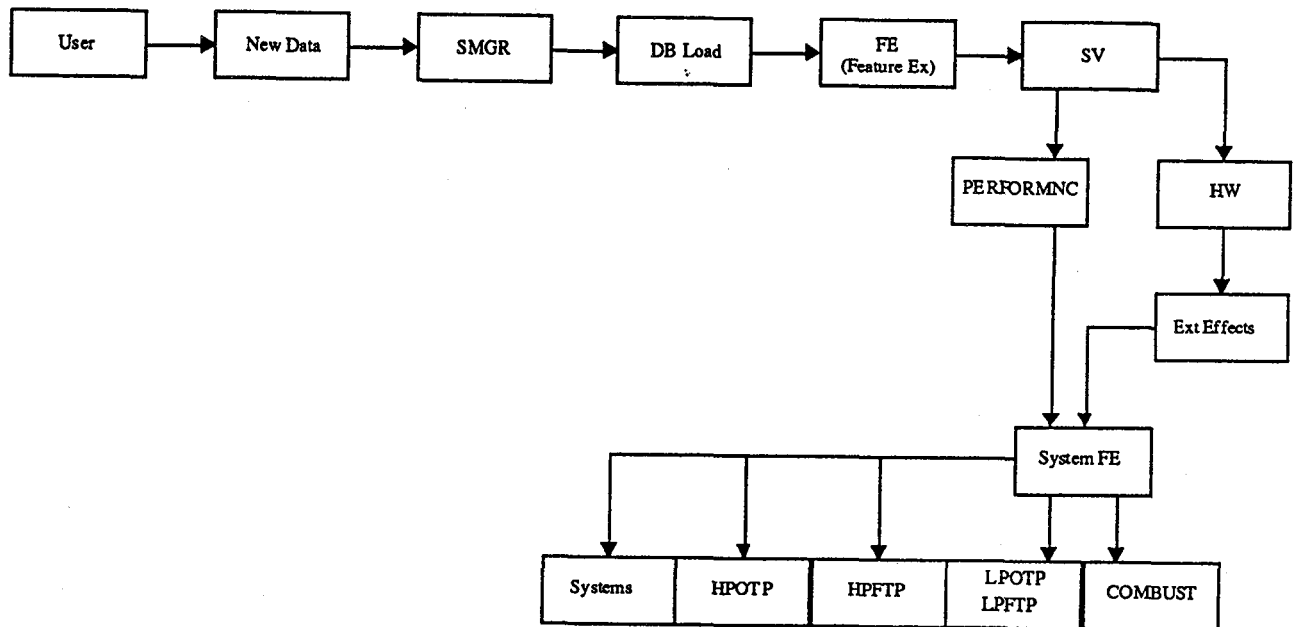


Figure 6. PTDS Execution Sequence

The following two versions of the feature extractor are used in each run of the PTDS:

FE — During the first invocation of the Feature Extractor, it looks at the raw data from the current test and computes features for use by the Sensor Validation module and the various component modules (i.e., the Turbomachinery and Combustion Devices modules) which look for combinations of features as indicators of anomalies.

SYSTEM FE — The second run of the Feature Extractor is specifically for the Systems Section. During this run, it looks exclusively at the normalized data for the current test produced by the Hardware Change and External Effects modules along with data from the comparison test, and computes Delta Level Shift features which are indicative of anomalous changes in engine behavior in the current test relative to the comparison test. This is performed in a separate run from FE1, because the External Effects module uses the results of Sensor Validation to determine which sensors to use in its normalization (see Figures 5 and 6).

Feature	Applicability	Description
Bistability	HPOTP	Used exclusively by the HPOTP module, this checks for preburner boost pump bistability on Rocketdyne HPOTPs.
Constant Thrust Different Than	Generic	Determines time intervals of constant power level.
Delta Different Than	Generic	Determines if two signals are significantly "different" from each other.
Drift	Generic	Given four signals—A1,A2,B1,B2—determines if A1-A2 is significantly different from B1-B2.
Erratic	Generic	Detects a slow linear change in a signal over a large time period.
Fit Line	Generic	Detects a non-smooth signal, by first fitting a first or second-degree polynomial to it (whichever fits better), then thresholding on the standard deviation of the fit.
Delta Fit Line	Generic	Fits a line to a signal over a specified interval of time.
Flat Signal	Generic	Fits a line to the difference between two signals.
Level Shift	Generic	Determines if a signal is "flat" over a specified time interval. Used primarily by sensor validation to detect sensors which are not connected to the engine during a test.
Delta Level Shift	Generic	Detects step shifts in a signal.
Limit Check	Generic	Detects step shifts in the difference between two signals.
Delta Limit Check	Generic	Detects excursions beyond a specified limit.
Noise	Generic	Detects when two signals vary from each other by more or less than a specified limit.
Nose Seal Leakage	HPOTP	Detects excessive noise, by computing standard deviations during one-second intervals and comparing them against a threshold.
Peak	Generic	Detects nose seal leaks on Rocketdyne HPOTPs, which are indicated by very subtle shifts down and back up in HPOTP intermediate seal purge pressure.
Piece-wise Linear	Generic	Detects excursions and recovery over a relatively long time interval which form the shape of a hill or peak in the signal.
Delta Piece-wise Linear	Generic	Creates a piece-wise linear model of a signal.
Redundant Channel Check	Generic	Creates a piece-wise linear model of the difference between two signals.
Spike	Generic	Detects significant deviations between data from redundant sensors by comparing the difference between N-point averages against a threshold.
Statistics	Generic	Detects rapid excursion and recovery (on the order of a few data samples) indicative of a "spiking" signal.
Delta Statistics	Generic	Computes mean, standard deviation, minimum and maximum values of a signal over a specified time interval.
Zero Shift Check	Generic	Computes statistics for the difference between two signals.
		Computes an average over a specified time interval and compares it to lower and upper limits. Used to detect failed sensors prior to engine start.

Table 1. Features Currently Detected by Feature Extractor Module

II.2. Sensor Validation Module

The Sensor Validation module is responsible for detecting instrumentation failures and anomalies. The results are both for display to the data analysts and for use by other PTDS modules which need to know which signals consist of valid data and which will yield erroneous analysis results.

The Sensor Validation module employs several strategies to detect instrumentation problems:

- First, a determination is made to see if a sensor is even available in the data file (a sensor will sometimes be dropped from a particular test).
- Checks are performed to see if a sensor is either flat for the duration of the test (indicating it is not hooked up to the engine) or is excessively noisy (indicating a potential electrical problem).
- The value of a sensor just prior to engine start is compared against known ambient conditions, and the sensor is failed if it is too far out of tolerance.
- Reasonableness checks are made on sensors during the firing to catch "hard" failures in which the sensor suddenly goes off-scale high or low.
- For sensors with redundant channels, a check is made between all the channels to ensure that they all stay within a tight tolerance of each other.
- Finally, an analysis is made of all features computed during the run of the feature extractor (FE) to determine if a feature found in one signal is also present in related signals. To do this, a map of related parameters is used which was derived through interviews with the data analysts (see Table 2). The result of this step is a preference ranking of the sensors which passed the first five screenings described above.

Map	Related Parameters
1	MCC CL DS P, LPFT IN P, LPFP SP, HPFP DS P
2	LPFP SP, HPFP IN P, LPFT IN P, FL PR INT P
3	MCC CL DS T, MCC CL D P, LPFP SP, HPFP DS T
4	FPB PC, FPOV POS, HPFT DS T, HPFP SP, PBP DS P
5	HPFP SP, HPFP DS P, HPFT DS T, FPB PC
6	MCC O INJ T, PBP DS T, ENG O IN T
7	ENG FL IN P, FAC FL FM DS P
8	FPOV POS, PBP DS P, HPFT DS T, FPB PC, HPFP SP
9	FPB PC, HPFT DS T, FPOV POS
10	HPFP IN P, LPFP SP, ENG FL IN P
11	OPOV POS, PBP DS T, HPOT DS T, OPB PC
12	OPB PC, HPOT DS T, OPOV POS, PBP DS P, MCC HG IN P
13	HPFP IN T, ENG FL IN T, HPFP DS T, LPFP SP
14	HPOP IN P, LPOP SP, ENG O IN P
15	ENG O IN T, FAC O FM DS T, MCC O INJ T, PBP DS T
16	ENG O IN P, FAC O FM DS P
17	ENG FL IN T, FAC FL FM DS T
18	HPOP SP, LPOP SP, PBP DS P, HPOP DS P

Table 2. Related Parameter Maps Used in Sensor Validation

II.3. Hardware Change Module

The Hardware Change module identifies changes in major SSME components between the current and comparison tests and generates a table of corrections applicable to the current

test data to account for differences in component performance whenever possible. In all cases, significant changes in hardware configuration are reported to the analysts along with the conclusions reached by the Systems Section. Table 3. shows the changes in hardware configurations which are recognized, and those which result in correction factors being applied to the current test data.

Component	Descriptive Parameters	Correction Applied?
Engine	Serial Number	
Powerhead	Serial Number	
Main Injector	Serial Number	
MCC	Serial Number	
Nozzle	Serial Number	
Controller	Serial Number	
Flowmeter	Serial Number	
FPOV, OPOV, MOV, MFV, CCV	Serial Number	
HPFD	Serial Number, Type	✓
F7 Orifice	Diameter	✓
HEX Orifice	Diameter	✓
HPOTP	Serial Number, TEM, PEM, TFPM, PHCM	✓
HPFTP	Serial Number, TEM, PEM, TFPM, PHCM	✓
LPFTP	Serial Number, TEM, PEM, TFPM, PHCM	✓
LPOTP	Serial Number, TEM, PEM, PHCM	✓

Table 3. Hardware Changes Recognized

II.4. External Effects Module

The External Effects module normalizes a number of dependent sensors in the current SSME test data with respect to a set of independent sensors describing engine-external conditions. The independent variables are: MCC PC, LPFP inlet pressure, LPOP inlet pressure, LPFP inlet temperature, LPOP inlet temperature, and facility mixture ratio. For each of these parameters a curve has been developed (using runs of the SSME Power Balance Model) which normalize all dependent parameters to the same inlet conditions found on the comparison test. The dependent parameters currently normalized are shown in Table 4.

II.5. Case-Based Reasoner Module

Level shift features found in the difference between the normalized data for the current test and the comparison test data by the systems feature extractor represent true anomalies which must be explained by the Case-Based Reasoner module. This module has three main components: a "comparator" which partitions the current test up into time intervals to diagnose; a "case base" or library of possible anomalies; and a case-indexing mechanism which finds a ranked list of candidates in the case base which best explain each anomaly.

PIDs	Parameter
2	HPOTP Speed
203, 204	HPFP Inlet Temp
17	MCC Coolant Discharge Pressure
18	MCC Coolant Discharge Temp
21, 595	MCC Oxid Injection Temp
24, 367, 371	MCC Hot Gas Injection Pressure
30, 734	LPOTP Speed
38, 139	MOV Position
52, 459	HPFP Discharge Pressure
58, 410	FPB Pc
59, 341	PBP Discharge Pressure
203, 204	HPFP Inlet Pressure
90, 334	HPOP Discharge Pressure
93, 94	PBP Discharge Temperature
40, 141	OPOV Position
42, 143	FPOV Position
209, 210	HPOP Inlet Pressure
231, 232	HPFT Discharge Temperature
233, 234	HPOTP Discharge Temperature
260, 261, 764	HPFTP Speed
436	LPFT Inlet Pressure
480	OPB Pc
659	HPFP Discharge Temperature
835	Fuel Press Interface Pressure

Table 4. Parameters Normalized by the External Effects Module

Each case represents the shifts, or gains, in a set of engine parameters expected to accompany a specific anomaly, along with the magnitude of the anomaly. These gains can be derived heuristically or from runs of the SSME Power Balance Model. The full set of cases is present in Section III, but the following example is for Piston Ring Seal Shift:

Parameter	Gain
LPFP SPD	100
FL PR INT P	-10
LPFT DeltaP	10

This case states that of the parameters given in Table 4, there are only significant changes in three parameters for a piston ring seal shift ($LPFT\ Delta P = LPFT\ IN\ P - LPFT\ DS\ P$, and $FL\ PR\ INT\ P = LPFT\ DS\ P$), and that the relative direction and magnitude of the shifts should be approximately proportional to the ones given in the above case.

The case-indexing mechanism compares each anomaly response pattern in the case base to the normalized delta level shifts found by System FE in order to select a small set of most probable cases. To perform this selection, the Case-Based Reasoner employs two indexing techniques: a sign or direction comparison and a case magnitude evaluation.

The first technique compares the directions of the observed gains with the directions of the gains expected for each hypothesis case in the case base. A score is generated for each observed gain and accumulated for a total case score. Table 5 defines the types of results

currently available, along with the score for each type. The accumulated score is used to rank the hypothesis cases for further evaluation. This provides an initial screening of the hypothesis case base. This screening reduces the processing time, by reducing the number of cases which undergo the computationally more intensive case magnitude evaluation.

Type	Description	Score
Match	Case Gain Matches Observed Gain	0
Not Covered	Observed Gain Not In Case Fact	1
Not Observed	Case Gain Not Observed	100/K
Opposite	Case Gain Opposite In Direction to Observed Gain	1000/K

Table 5. Direction Comparison Types
The variable K found in the Score column is equal to the number of parameter shifts in the particular case being evaluated.

The case magnitude evaluation is performed on each hypothesis case selected by the sign comparison technique. Given case i with anomaly magnitude M_i and gains C_{ij} (for each engine parameter j) and an observed shift in engine parameters described by the gains O_j , scale factors are computed for each $C_{ij} \neq 0$:

$$S_{ij} = O_j / C_{ij}$$

The mean (μ_i) and standard deviation (σ_i) of the set of scale factors for each case is the computed. The cases are ranked according to minimum standard deviation in scale factors (those with lower standard deviations provide a better overall fit of the gains in the case to the observed gains, taking a linear scaling of the magnitude of the anomaly into account). Given the best case (the one with the minimum σ_i), the estimated magnitude of the anomaly is given by $\mu_i * M_i$. The set of best matches are then output to the database along with a description of plots which support their selection.

II.5. Performance Module

The Performance Module essentially runs part of the SSME Power Balance model at one-second intervals during steady-state conditions to determine the following parameters:

LPFP Efficiency, LPFT Efficiency
HPFP Efficiency, HPFT Efficiency
LPOP Efficiency, LPOT Efficiency
HPOP Efficiency, HPOT Efficiency, PBP Efficiency

A simplified Level Shift routine is then run to detect significant shifts in any of these parameters. Results are then posted to the database for later viewing by the analysts.

III. Anomalies Currently Detected by the Systems Section

In addition to detecting instrumentation failures (in the Sensor Validation module), pump efficiency shifts (in the Performance module) and noting changes in hardware configuration (in the Hardware Change module), the following cases have been developed for Case-Based Reasoner by runs of the SSME Power Balance Model. At this time, however, only the cases for positive and negative piston ring seal shift have been validated by analysts and tested against SSME test datasets. Validation of the remaining cases and development of new ones is currently being performed by personnel at LeRC and MSFC.

Case	Parameter	Gain
Primary Piston Ring Seal Shift (Close)	LPFP SPD	+100
	FL PR INT P	-10
	LPFT DeltaP	+10
Primary Piston Ring Seal Shift (Open)	LPFP SPD	-100
	FL PR INT P	+10
	LPFT DeltaP	-10
MCC Pc Biased Hi 20 psi	HPFP DS P	-15
	HPOP SPD	-99
	HPOP DS P	-18
	OPOV	-0.3
	FPB Pc	-14
	OPB Pc	-22
	HPOT DS T	-18.3
MCC PC Biased Low 20 psi	HPFP DS P	+15
	HPOP SPD	+100
	HPOP DS P	+18
	OPOV	+0.4
	FPB Pc	+14
	OPB Pc	+22
	HPOT DS T	+18.2
Eng Fuel Flowmeter Biased Hi 3lbm/sec	HPFP SPD	-275
	HPFP DS P	-58
	HPOP SPD	+178
	HPOP DS P	+20
	OPOV	+1.1
	FPOV	-2.2
	FPB Pc	-41
	HPOT DS T	102.9
Eng Fuel Flowmeter Biased Lo 3lbm/sec	HPFP SPD	+294
	HPFP DS P	+60
	HPOP SPD	-162
	HPOP DS P	-19
	OPOV	-1
	FPOV	+2.8
	FPB Pc	+43
	HPOT DS T	-90.4
MCC Combustion Eff Decrease 1 sec ISP	HPOP SPD	+60
	HPOT DS T	+10.6

Case	Parameter	Gain
Fuel Leak 3lb/s	HPFP IN P	-42
	HPFP DS P	-26
	HPFP DS T	+23.1
	HPOP IN P	-2.3
	HPOP SPD	+176
	HPOP DS P	+17
	OPOV	+0.7
	FPOV	-0.5
	HPOT DS T	+57.7
Nozzle Coolant Leak 4m/s	HPFP SPD	-74
	HPFP DS P	-42
	HPFP DS T	+32.7
	HPOP SPD	+230
	HPOP DS P	+29
	HPOT DS T	+97.1
Ox Leak at PBP Discharge 9lbm/s	HPOP IN P	-4.3
HPFP In T Biased Hi 2deg	HPFP SPD	+294
	HPFP DS P	+60
	HPOP SPD	-162
	HPOP DS P	-19
	OPOV	-1
	FPOV	+2.8
	FPB Pc	+43
	HPOT DS T	-90.4
HPFP In T Biased Lo 2 deg	HPFP SPD	-275
	HPFP DS P	-58
	HPOP SPD	+178
	HPOP DS P	+20
	OPOV	+1.1
	FPOV	-2.2
	FPB Pc	-41
	HPOT DS T	+102.9
LPFT Bearing Cool Inc 1lbm/s	HPFP IN P	-8
	HPFP SPD	+820
	HPFP DS P	+41
	HPFP DS T	+48.1
	HPOP SPD	+51
	FPOV	+2.1
	FPB Pc	+45
	OPB Pc	+15
	HPOT DS T	-21.1
CCV Resistance Inc 10%	HPFP IN P	+26
	HPFP SPD	+422
	HPFP DS P	+235
	HPFP DS T	+80.8
	FPOV	+2.9
	FPB Pc	+33
MFV Resistance Inc 4%	HPFP SPD	+147
	HPFP DS P	+75
	FPOV	+0.9
	HPOT DS T	-8.7

Case	Parameter	Gain
HPOP Disch Resistance Inc 15 posts	HPOP SPD	+208
	HPOP DS P	+46
	FPOV	-1.3
	OPB Pc	+25
	HPOT DS T	+38.5
MOV Resistance Inc 98%	HPOP IN P	+6.4
	HPOP SPD	+88
	HPOP DS P	+53
	FPOV	-1
	OPB Pc	+12
HPFP In P Inc 25psi	HPOT DS T	+20.2
	HPFP IN P	+24
	HPFP DS P	+26
	HPFP DS T	-23.1
	HPOT DS T	-22.1
HPFP In P Dec (LPFP Cause) 25psi	HPFP IN P	-24
	HPFP SPD	+67
HPFP In P Dec (LPFT Cause) 25psi	HPFP IN P	-24
	HPFP DS P	-18
	HPFP DS T	+18.3
	HPOT DS T	+16.4
Fuel In Temp Inc 3deg	HPFP SPD	+609
	HPFP DS P	+27
	HPFP DS T	+37.5
	FPOV	+1.9
	FPB Pc	+34
	HPOT DS T	-24
LOX In Temp Inc 7deg	HPOP SPD	+415
	HPOP DS P	+14
	OPOV	+1.1
	OPB Pc	+26
	HPOT DS T	+38.5
LPOT Flow Inc 20lbs	HPOP IN P	+32.3
	HPOP SPD	+162
	OPOV	+0.5
	FPOV	-0.5
	OPB Pc	+17
	HPOT DS T	+26
LPOT Flow Dec 20lbs	HPOP IN P	-35.5
	OPOV	-0.2
	FPOV	+0.4
	HPOT DS T	-14.4
HPFT Efficiency Dec 5%	HPFP IN P	+5
	HPFP SPD	+95
	HPFP DS P	+53
	HPFP DS T	+120.2
	FPOV	+7.2
	FPB Pc	+85
	OPB Pc	+18
	HPOT DS T	-61.5

Case	Parameter	Gain
HPFP Efficiency Decrease 5%	HPFP IN P	+9
	HPFP SPD	+147
	HPFP DS P	+72
	HPFP DS T	+133.7
	HPOP IN P	-1.4
	HPOP DS P	-12
	OPOV	-0.2
	FPOV	+9.6
	FPB Pc	+102
	OPB Pc	+23
	HPOT DS T	-71.2
HPOT Efficiency Decrease 5%	HPFP IN P	+3
	HPFP SPD	+45
	HPFP DS P	+25
	HPFP DS T	-25
	OPOV	+4.5
	FPOV	+0.4
	OPB Pc	+65
	HPOT DS T	+115.4
HPOP Efficiency Decrease 5%	HPFP IN P	+4
	HPFP SPD	+67
	HPFP DS P	+38
	HPFP DS T	-38.5
	HPOP IN P	+1.5
	OPOV	+7.9
	FPOV	+1
	OPB Pc	+97
	HPOT DS T	+167.3
PBP Efficiency Decrease 5%	OPOV	+0.3

IV. Conclusion

The SSME Post-Test Diagnostic System is not intended to replace the human data analysts, but rather is intended to be used as an additional cross-check for data obtained from each test firing. The PTDS should provide a standardized set of analyses which always look at each test in the same manner, and has the potential for detecting very subtle anomalies that analysts might otherwise overlook (this has already been demonstrated on the HPOTP module).

In addition to its immediate applicability to the SSME, the PTDS has been designed as a generic system which could be applied to the analysis of test data from other rocket engines. Elements of the PTDS have already been applied to the analysis of Atlas/Centaur data, and an application for Titan data analysis is planned.

Automated checkout procedures such as those implemented in the PTDS are crucial elements of integrated vehicle health management systems whose goals are to reduce operations costs and turnaround times, and increase reliability. Health management tools such as the PTDS are expected to be requisite elements of most future rocket propulsion systems.

IV.1. Future Work

Although the PTDS now detects and diagnoses a broad range of SSME anomalies, there are several areas in which it could be further enhanced, including:

- **Combustion Devices Analysis** — The Combustion Devices group in the MSFC Propulsion Branch looks specifically at the performance of and anomalies in the two SSME preburners and the main combustion chamber. A PTDS module is currently under development which encodes their procedures for detecting common anomalies in these components.
- **Turbomachinery Analysis** — The HPOTP module only looks for anomalies in the SSME HPOTP; there are three other pumps on the engine which are currently not covered by the PTDS. Development of modules to cover the analysis of data from the other three turbopumps (HPFTP, LPOTP, and LPFTP) is planned for 1995.
- **Transient Analysis** — The PTDS currently performs steady-state (constant power level) analyses only. However, several significant anomalies occur during the startup and shutdown transients. A module which specifically looks at SSME data during these times is currently under development.
- **Flight Data Analysis** — The PTDS is currently configured to analyze ground test data only. Several extensions to the system would need to be made to enable it to analyze flight data, and new, flight-specific anomalies may need to be added to the various diagnostic modules. This work is currently scheduled for early 1996.
- **Integration with MSFC Plot Program** — The PTDS graphical user interface currently displays plots of engine data for the anomalies it has detected. The MSFC data analysts have been using a PV~Wave-based plotting program for the last several years which supports a wide range of scaling, zooming, and cross-plotting options. Ultimately, the PTDS should utilize this program to display its plots, since the analysts are already familiar with its functionality and it is more flexible than the PTDS plotting routine. This integration task is scheduled for 1995.

- **Dynamics Data Analysis** — Several significant anomalies, such as bearing wear, can only be detected through analysis of accelerometer data. Currently, this data is reviewed by a separate group at MSFC, which is in the process of automating their data analysis procedures. It should be possible in the future to automatically integrate the results of this system with the PTDS to obtain an overall best picture of the health of the engine.
- **Automated Case Entry** — Anomaly cases for the Case-Based Reasoner must currently be entered by hand via the TekBase database graphical user interface (KingFisher). A more automated means should be provided for case entry, which ideally would take a diagnosed and validated anomaly from a test dataset, extract the appropriate gains, and make an entry into the case base at the push of a button.
- **Display and Modify Intermediate Results** — Currently, the PTDS analyzes the data from a test for several hours and then displays its results. Although some of the intermediate decisions reached (e.g., detection of bad instrumentation, hardware changes, etc.) are available for review by the analyst, there is no capability to modify any of these decisions. Thus, if the system makes an incorrect decision at an early stage of its reasoning (e.g., not disqualifying a failed sensor) then all results based on this decision will be incorrect. The PTDS could be made more interactive by allowing analysts to incrementally run each of the modules and then review and possibly correct its results before continuing on with the analysis.
- **Analyst Entry of Diagnostic Rules** — Many of the diagnostic rules used in the PTDS are very simple in form (e.g., many of them just perform limit checks on signals). A graphical template (or "form") could be constructed for each of these rule types which would allow analysts to enter or modify these classes of diagnostic rules. This would give the analysts a better understanding of how the system works, and enable them to take over some part of the maintenance of the system.
- **Indexing of Similar Anomalies** — When the PTDS is displaying an anomaly, provide a button which will automatically show a list of all previous tests which had the same anomaly and allow the analyst to quickly plot the data for comparison against the current test.
- **Extend the Classes of Events the Systems Section Detects** — Currently, the Systems Section only responds to level shifts in the difference between the current test and a comparison test. Often, anomalies will present themselves as spikes, peaks, or drifts in the difference signal. The Systems Section should be extended to detect and diagnose these events (although, the Case-Based Reasoner may need additional cases to accommodate the dynamic effects of rapid spikes or peaks).
- **Hybrid Queries Across TekBase and SSME DataFile** — MSFC data analysts spend a lot of time looking for data from past tests which meet specific criteria, such as "find all previous tests on stand A1 which had HPOTP #123 and ran for at least 30 seconds at 109% at minimum LOX NPSP". Currently, information about the hardware configuration (such as HPOTP serial numbers) and general test profile are stored in the TekBase database, while more specific information about the test data (such as time at 109% and minimum LOX NPSP) must be computed from data stored in the test data file. A hybrid query mechanism could be developed which could answer queries such as the one above, by satisfying as much as the query as possible against the database, then automatically going to the data files and computing the remaining information required.

Acknowledgements

The PTDS has been developed over the last five years by a large development team. The members, past and present, include:

Rick Ballard
Tim Bickmore
Jeff Cornelius
J. Allen Crider
Chris Fulton
Mark Gage
Amy Jankovsky
Bill Maul
Catherine McLeod
Claudia Meyer
Pam Surko
Virginia Tickle
Luis Trevino
Jean Tucker
June Zakrasjek

Many data analysts at MSFC have contributed to the PTDS by answering our endless stream of questions about SSME data analysis. These analysts include:

Glenn Doughty
Bill Foster
Dave Foust
Darrel Gaddy
Taylor Hooper
Randy Hurt
Mike Kynard
Larry Leopard
Lewis Maddux
Marc Neely
Brian Piekarski
Eric Sanders
Chris Singer
Dave Vaughan
Glenn Wilmer

Finally, this work could obviously not have been completed without the support of the project managers and supervisors:

June Zakrasjek	NASA LeRC
Catherine McLeod	NASA MSFC
Dave Seymour	NASA MSFC
Randy Bickford	Aerojet
Roy Michel	Aerojet

References

- ¹Zakrasjek, June, *The Development of a Post-Test Diagnostic System for Rocket Engines*, AIAA-91-2528, AIAA/SAE/ASME/ASEE 27th Joint Propulsion Conference, June 1991.
- ²Bickmore, Timothy W., and Maul, William A., "A Qualitative Approach to Systemic Diagnosis of the SSME," *AIAA Aerospace Sciences Conference and Exhibition*, Reno, Nevada, January 1993.
- ³Bickmore, Timothy W., *SSME HPOTP Post-Test Diagnostic System Enhancement Project*, NASA Contractor Report 4643, Contract NAS3-25883, January 1995.,

SSME Post-Test Diagnostic System Systems Section

**Final Report
Attachment #1**

User's Guide

Post-Test

Diagnostic System

(PTDS)

User's Guide

Prepared by:

J. Allen Crider

Computer Sciences Corporation

16 February, 1995

Preceding Page Blank

Table of Contents

Table of Contents	i
List of Figures	ii
Acknowledgments	iii
I. Introduction	1
System Requirements	1
Prerequisites	1
Execution Modes	2
II. PTDS Execution	3
Hardware Configuration Data Entry	3
Test Data Analysis	3
Graphical Review of Results	5
Anomaly Database Update and Review	13
III. Maintaining the Historical Database	19
Updating the Historical Database	19
Viewing the Historical Database	19

Preceding Page Blank

List of Figures

Figure 1. Main panel for the program new_data.....	4
Figure 2. Options pull-down menu for the program new_data.....	4
Figure 3. PID Override panel for the program new_data.....	5
Figure 4. Main panel for the program ehms.....	6
Figure 5. Panel for the "History" option from the "Analysis Tools" menu.....	7
Figure 6. "History" panel with "LRU's" pull-down menu.....	8
Figure 7. HPOTP window.	9
Figure 8. Systems window.	11
Figure 9. "Make Pids" window for modifying PID buttons on plant diagrams.....	11
Figure 10. Explanations window for anomalies.....	12
Figure 11. Plot panel for program ehms.	13
Figure 12. Main panel for the program anom.....	14
Figure 13. "Fixed Fields" and "Anomaly Title" areas of the Anomaly Database query window after selecting "LRU" from the "Location" menu button.....	15
Figure 14. "Anomaly Title" area of the Anomaly Database query window after selecting "System" from the "Location" menu button.....	15
Figure 15. Selection panel for anom displayed when more than one anomaly satisfies a query or the "Read Selection" button on the main panel is pressed.....	16
Figure 16. Main panel for the program anom after a data retrieval.....	17
Figure 17. Main panel for the program anom when using the "Add" command.....	18

Acknowledgments

This User's Guide and Programmer's Guide contains contributions from various members of the Post-Test Diagnostic System (PTDS) development team. The members of the development team are

Rick Ballard

Tim Bickmore

J. Allen Crider

Chris Fulton

Bill Maul

Catherine McLeod

Claudia Meyer

Virginia Tickle

Luis Trevino

June Zakrajsek

I. Introduction

This manual describes the operation of the Post-Test Diagnostic System (PTDS), which detects and diagnoses anomalies in the Space Shuttle Main Engine (SSME) by analyzing data from ground tests of the engine.

The PTDS is an expert system which utilizes heuristics and case based reasoning techniques to identify common anomalies and observations in the data. It operates by first running "feature extraction" routines to scan the raw test data and identify any features of interest, such as spikes, shifts, peaks, or limit violations. Expert system rules then analyze combinations of these features which are indicative of known anomalies. Results from the system are stored in a relational database for future reference, and can be viewed via a graphical user interface which displays observations and supporting plots of test data for a selected test of interest.

From the viewpoint of the average user, two primary programs comprise PTDS: `new_data` and `ehms` (engine health management system). `new_data` allows a user to run the diagnostic system on a new test and `ehms` allows the user to view the results of the diagnostics using graphical and textual screens.

The PTDS has been designed to validate sensors, required for the analysis, and to incrementally add modules to diagnose Line Replaceable Units (LRUs) and the overall system. Currently, PTDS is composed of a SYSTEMS module, which analyzes overall engine system conditions, and a HPOTP module which identifies approximately 80 HPOTP anomalies.

System Requirements

The PTDS currently runs on Sun SPARCstations, and requires the following commercial software packages:

- Sun Operating System SunOS 4.1.3
- X Window System
- Motif
- TekBase Relational Database Management System
- PV-Wave Command Language

Full installation and setup of this system is beyond the scope of this manual. Please see your system administrator for details.

Prerequisites

To use the Post-Test Diagnostic System, you should already be familiar with the following:

- Basic Unix commands (see *SunOS User's Guide: Getting Started* or other Unix references)
- Use of a windows-based graphical user interface (see *OSF/Motif User's Guide*)
- Ability to view and update tables in TekBase (see *Kingfisher Users Guide*)

In order to run the programs which make up the Post-Test Diagnostic System, you will need to have the following environment variables defined (usually in your `.cshrc` file):

- `NASA_HOME` must be set to the directory containing the PTDS executables (e.g., on the Propulsion Lab network at MSFC use `/hd4/PTDS/bin`)

Preceding Page Blank

- `NASA_TEST_DATA` must be set to a colon-separated list of the directories in which SSME tests data is stored (e.g., on the Propulsion Lab network at MSFC use `/hd1:/hd2:/hd3:/data1:/data2:.`)
- `SYSTEMS_KBDir` must be set to the directory containing the CLIPS files for the SYSTEMS module (e.g., on the Propulsion Lab network at MSFC use `/hd4/PTDS/bin/system_kb`)
- `INCLUDE_PIDS_FILE` is required by `FEATURES` and `db_load` and must be set to the name of the file containing a list of all PIDs (Parameter IDs) for a test (e.g., on the Propulsion Lab network at MSFC use `include.pts`)
- `FL_RC_PATH` must be set to the directory containing the resource file `.fl1librc` if it is not located in a default location (e.g., on the Propulsion Lab network at MSFC use `/hd4/PTDS/data`)

The `NASA_HOME` directory should be added to your search path. Normally this can be done by adding the following line to your `.cshrc` file after the definition of `NASA_HOME`:

```
source $NASA_HOME/./data/.nasarc
```

This will also set all of the environment variables listed above other than `NASA_HOME`. Check with your system administrator if your installation is set up differently.

Your environment must also be configured for proper use of TekBase and PV~Wave. (For the Propulsion Lab network at MSFC, adding the following lines to your `.cshrc` file will be sufficient:

```
setenv TQL_SERVER_DIR /u/metrica4.0/tqlserver.SUN4
setenv TQL_CLIENT_DIR /u/metrica4.0/tqlclients
setenv PATH "$TQL_CLIENT_DIR/bin:$PATH"
setenv WAVE_DEVICE X
setenv WAVE_DIR /hd3/vni/wave
setenv WAVE_PATH $WAVE_DIR/lib
```

For those running PTDS on other systems, see your system administrator for the proper paths to use.)

Finally, the data files for the test you wish to analyze should be accessible from the machine you are running on. Currently, both compressed and uncompressed MSFC datafile formats are supported.

Execution Modes

The PTDS is configured to run as a "batch" process (typically run overnight) which analyzes all SSME components in parallel. In addition, each module can be run interactively. In interactive mode, only the component specified will be analyzed and all results are simply displayed in textual form. See "Test Data Analysis" in Section II for more information about running PTDS in either mode.

II. PTDS Execution

The full SSME Post-Test Diagnostic System is run in four steps: (1) hardware configuration data entry; (2) test data analysis; (3) graphical review of results; and (4) anomaly database update and review.

Hardware Configuration Data Entry

The data necessary to run the Post Test Diagnostic System (PTDS) are supplied by Rocketdyne, Canoga Park, through jetson in a comma delimited file approximately one day before the test firing. The data can be found in the directory /home/gyork.

Any changes to the pretest information are sent in a post-test file and the files are updated. Pump and turbine efficiency data is sent to NASA via jetson immediately after a test firing. However, in cases where a pump has been changed out, efficiency data is usually delayed. The pretest, post-test and efficiency data files are identified by test numbers and suffixes. (Examples are a10750.pre, a10750.pos, and a10750.eff, respectively). This data has to be manipulated before transferring into the database. To do so the user types:

```
hwconv infile outfile
```

where *infile* is the pretest and/or post-test file you want to access and *outfile* is the modified hardware file you want to create. (Example: hwconv a10750.pre a10750.hmod, where hmod is the suffix of the modified version of the hardware file.)

This procedure is also used to create the other files needed for PTDS access. Thus the user types:

```
infconv infile outfile
```

where *infile* is the pretest and/or post-test file you want to access and *outfile* is the modified test information file you want to create. (Example: infconv a10750.pre a10750.imod, where imod is the suffix of the modified version of the test-information data file.) And,

```
effconv infile outfile
```

where *infile* is the efficiency file you want to access and *outfile* is the modified efficiency file you want to create. (Example: effconv a10750.eff a10750.emod, where emod is the suffix of the modified version of the efficiency data file.)

Now that all the data has been modified, it is ready to be transferred into the TekBase SSME_DB database (see *Kingfisher Users Guide*).

The .hmod files transfer into the TST_HW table. The .imod files transfer into the TST_INFO table. The .emod files transfer into the TST_PERF table. (See Section 2.1 for Table Listings.)

This data is now ready to be accessed by the Post-Test Diagnostic System.

Test Data Analysis

To begin full PTDS analysis of a new SSME test, once the data files are on-line, run the program *new_data*. This program is used to launch the PTDS. It allows the user to specify the tests which are to be run by the system and to start the session manager on those tests, which in turn executes the other components of the system in a batch mode. Normally, *new_data* is the only program the user will need for the test data analysis step.

Usage:

new_data [&]

This command displays the main panel for the program as shown in Figure 1.

Options	Go	Help
Current Test:		Submit Date: 06/25/94
Previous Test:		

Figure 1. Main panel for the program new_data.

To launch PTDS from new_data, enter the Current Test ID in the text field labeled Current Test and enter the Comparison Test ID in the text field labeled Previous Test. Selecting Go from the menu bar will then queue the specified test(s) for the session manager, and if the session manager is not already running, it will begin the session manager.

The Options pull-down menu is shown in Figure 2. The Clear option is used to clear the Current Test and Previous Test text fields. The Exit option is used to exit the program. The PID (Parameter Identification) Override option displays the PID Override panel.

Options	Go	Help
PID Override		
Clear		Submit Date: 06/25/94
Exit		
Previous test:		

Figure 2. Options pull-down menu for the program new_data.

The PID Override panel, shown in Figure 3, is used to substitute different PIDs for the default PIDs during analysis of the test(s) currently indicated on the main panel. To override a PID, enter the default PID name in the left text item and the substituted PID name in the right text item. Then press the Add button. The substitution will be added to the scrolling list at the bottom of the panel. If a mistake is made, the item may be selected in the scrolling list and the Delete button used to remove it. The Clear button will delete all items from the list and the Close button closes the panel.

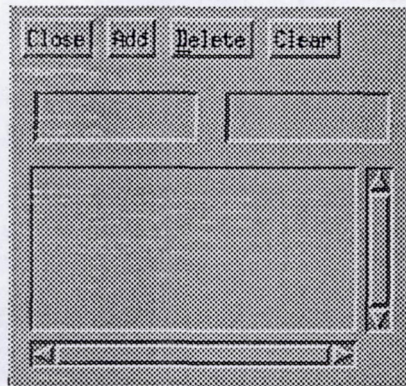


Figure 3. PID Override panel for the program new_data.

To use interactive mode to perform diagnosis on a Line Replaceable Unit of the SSME, simply type *component_interactive* at the Unix command line, where *component* is the name of the LRU that requires diagnosis, e.g., *HPOTP_interactive*. After a brief loading period, the program will ask you to enter the test ID for the current test. The test ID should be specified as a six-character string of the form A20551, A40134, etc. The program will then ask you to specify the test ID for a comparison test. You can either enter a test ID or simply a carriage return if a good comparison test is not available (most analyses can be run without the use of a comparison test). The program may also prompt the user for other information. For example, The HPOTP module will ask you if this test uses a Rocketdyne or Pratt & Whitney (ATD) pump, and finally, the program will ask you for the name of a log file to store the analysis results in. You can either enter a valid filename, or simply a carriage return to indicate that you do not want a log file created.

The program will then perform its analysis of the component, periodically printing results as they are obtained. At the end of its execution, the program will ask if you want to update the historical database with the parameters from the current test (just answer yes or no). Program execution varies according to the component.

See Attachment #3 of *SSME HPOTP Post-Test Diagnostic System Enhancement Project* for a sample session log for the HPOTP module.

Most of the other components of PTDS can also be run individually, either interactively or as background processes, although this should rarely be done under normal circumstances. Information on running the other programs individually is provided in Section 1 of the Programmer's Guide.

Graphical Review of Results

To view the results of a PTDS analysis, or to check the progress of an analysis in progress, type *ehms* at the Unix command line. This will bring up the window shown in Figure 4. The top scrolling window is the test status board which displays the status of all analyses in progress in its top portion (with a check mark showing which modules have completed; a clock icon denotes analysis still pending), and a list of all completed analyses at the bottom. To select a test to review, simply click on the test ID with the left mouse button. The system will then take a few minutes to load in the analysis results, and highlight any components on the SSME plant diagram which were found to be anomalous. To view the results of a component analysis, left-click on the desired component icon in the SSME plant diagram, to bring up a more detailed component schematic. Examples of this window are shown in Figures 7 and 8, which

display the HPOTP schematic and diagnostic results and the Systems schematic and diagnostic results, respectively.

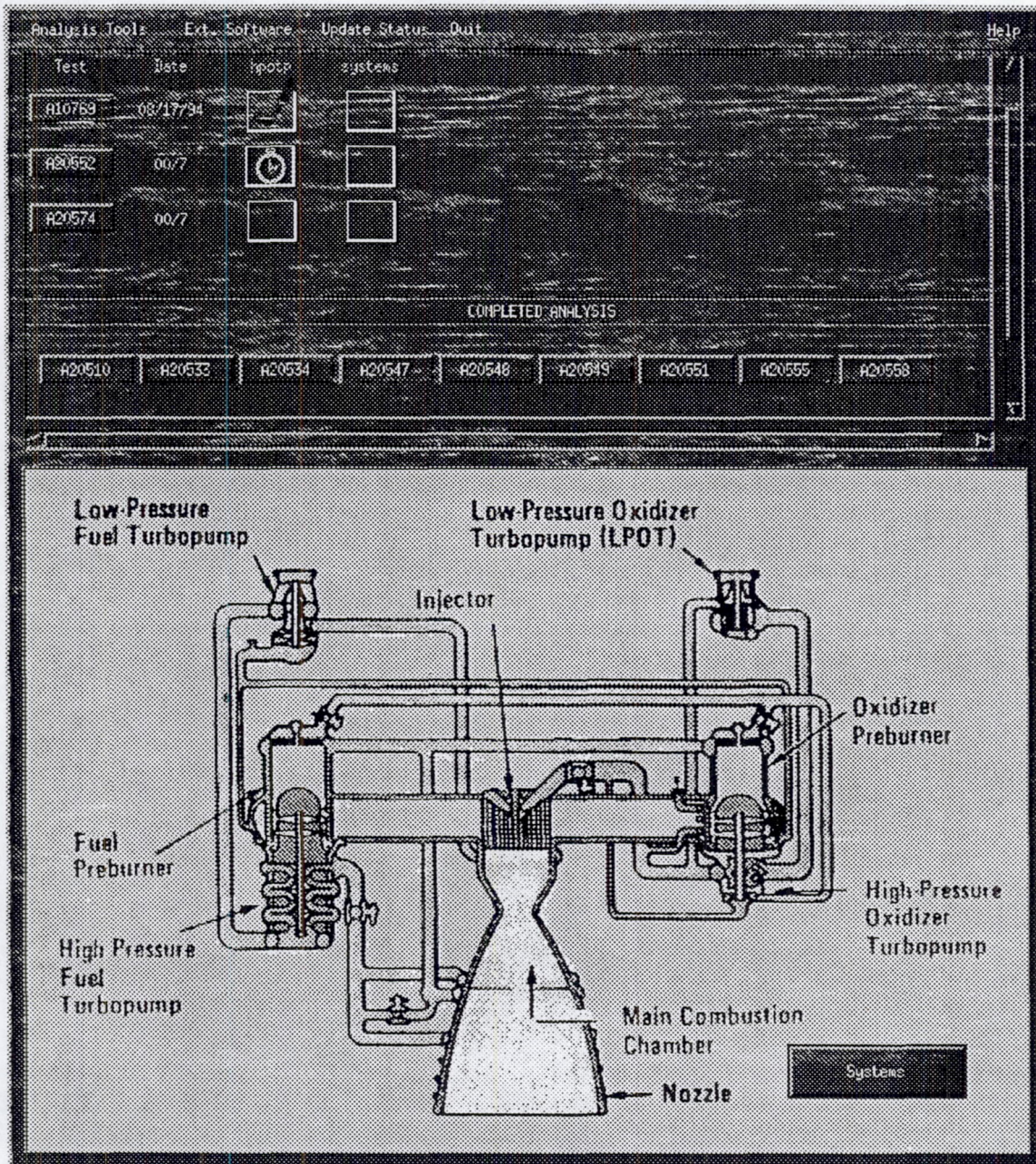


Figure 4. Main panel for the program ehms.

The menu bar at the top of the main panel provides a means to start execution of other programs useful in the analysis process. There is a pull-down menu under "Analysis Tools" with three options. The first option, "What If", is currently inactive. The second option, "History", brings up the panel shown in Figure 5. The last option, "Plot Pkg", starts the program sunplot in a new terminal window. "Ext. Software" starts other useful programs through another pull-down menu. The options on this menu are "new_data", which starts the program new_data (see "Test Data Analysis" above for usage

information), "Anomaly DB", which starts the program anom (see "Anomaly Database Update and Review" below for usage information), and "Kingfisher", which starts the program kingfisher (see *Kingfisher Users Guide* for more information). "Update Status" may be used to update the status area in the top scrolling window to show any modules which have completed since ehms was started. "Quit" exits the program and currently there is no information available through the "Help" button.

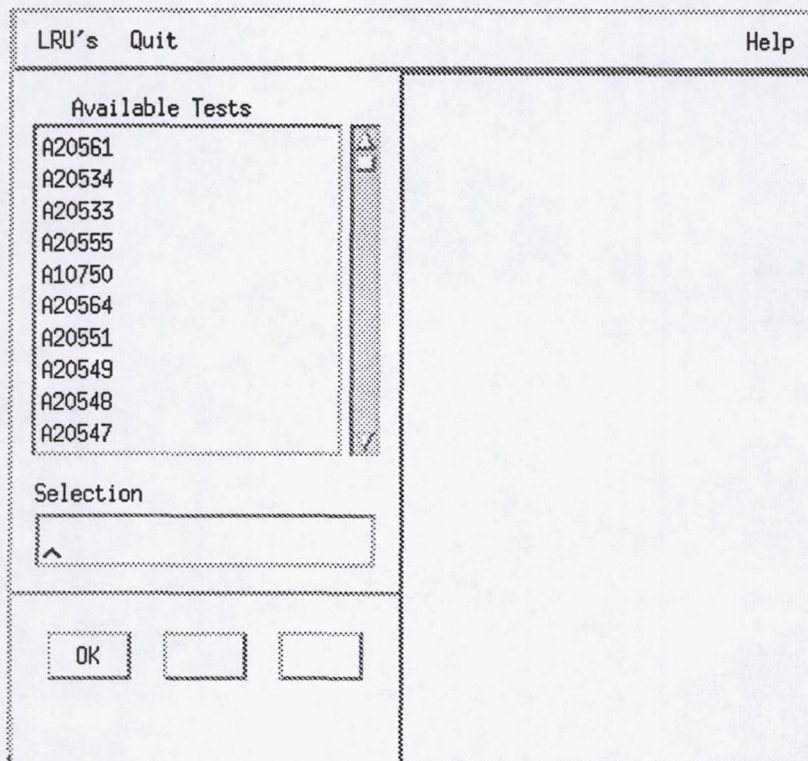


Figure 5. Panel for the "History" option from the "Analysis Tools" menu.

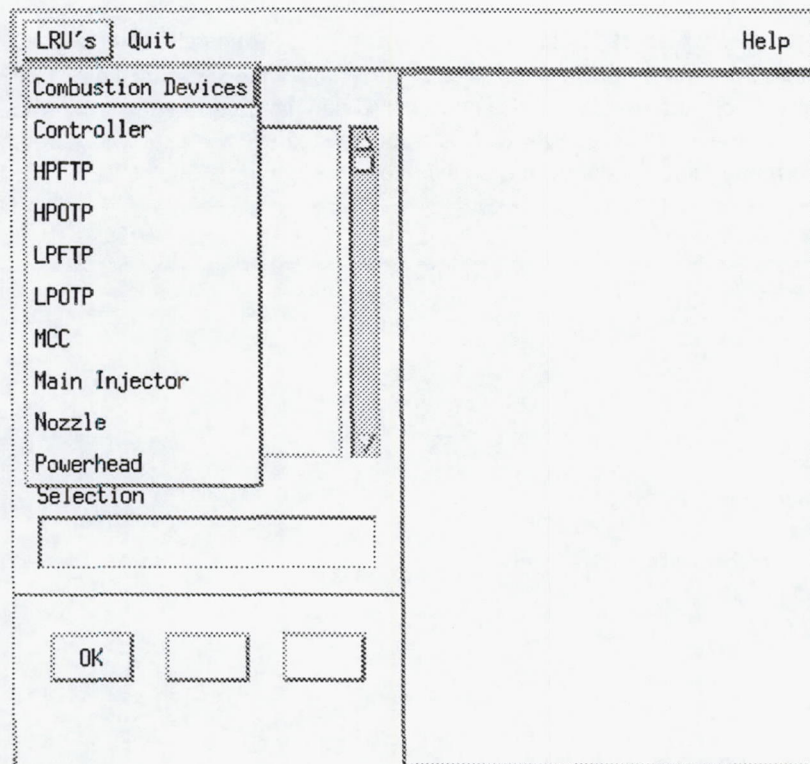


Figure 6. "History" panel with "LRU's" pull-down menu.

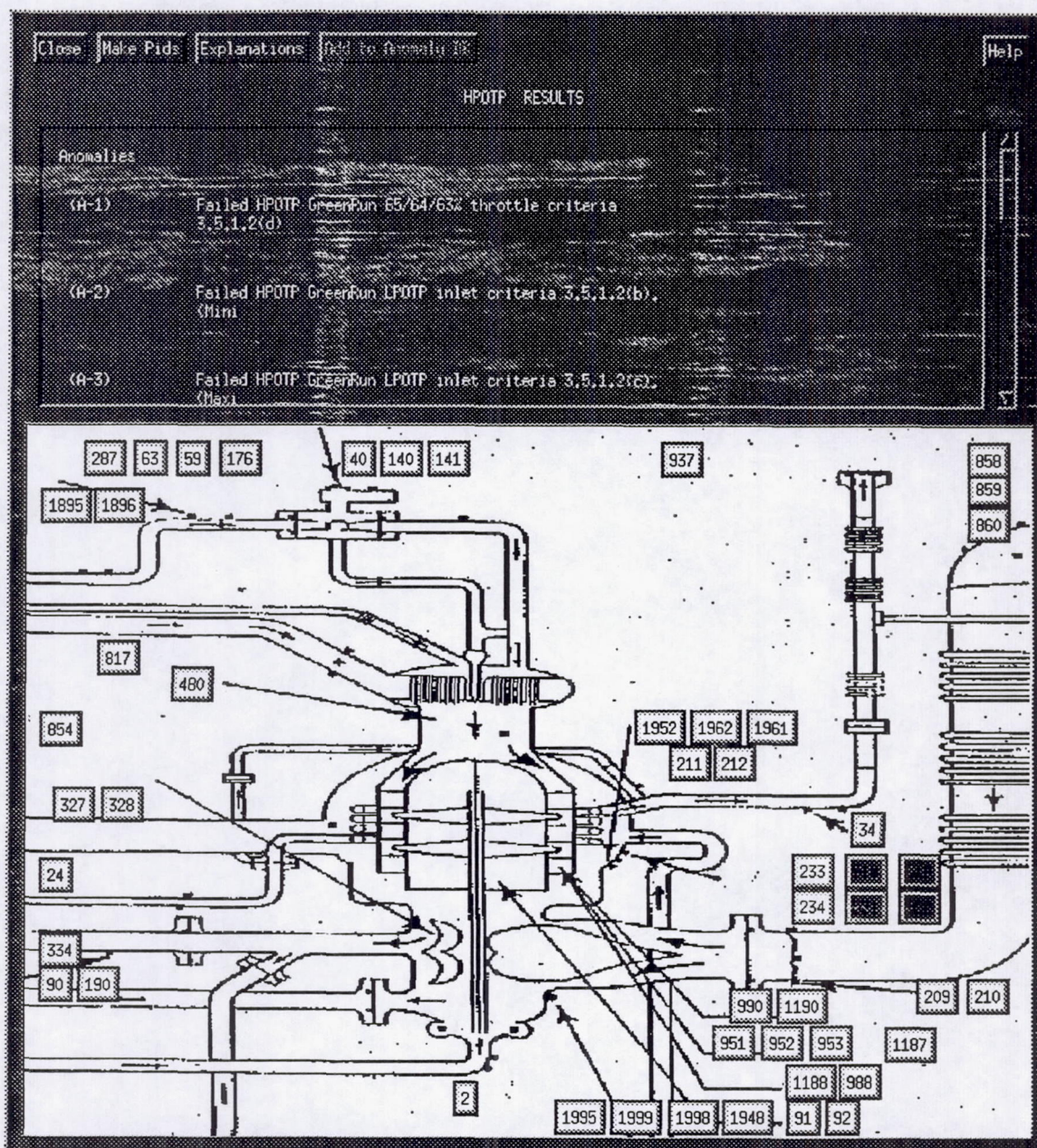


Figure 7. HPOTP window.

The top portion of the component window shows the list of ranked anomalies found for the selected test, broken down into three categories: anomalies, observations, and instrumentation. As shown in Figure 7, the bottom portion of the HPOTP window shows a plant diagram of the HPOTP, with buttons representing sensors used in analysis of the HPOTP (the buttons are labeled with the sensor's PID name). To view the raw data for any sensor, simply left-click on the corresponding button. To obtain plots of data which support an anomaly or observation, left-click on the text of the description. Similarly, the bottom portion of the Systems window contains the SSME plant diagram with buttons representing applicable sensors as shown in Figure 8.

The "Close" button at the top of each component window is used to close the window. The "Make Pids" button brings up the window shown in Figure 9, which may be used to update the buttons in the plant diagram at the bottom of the window. The "Next" and "Prev" buttons can be used to move through the list of PIDs for which buttons are included on the plant diagram. The current PID name will be displayed in the text item labeled "Pid name:" and the position of the button will be displayed in the text items labeled "x:" and "y:".

The "Explanations" button brings up the window shown in Figure 10. If an anomaly has been selected on the component window, this window will contain a list of the data which caused the anomaly to be identified by PTDS. If no anomaly has been selected, the scrollable portion of the window will be blank.

The "Add to Anomaly DB" button is only active if an anomaly is selected. Currently, no information is available through the "Help" button.

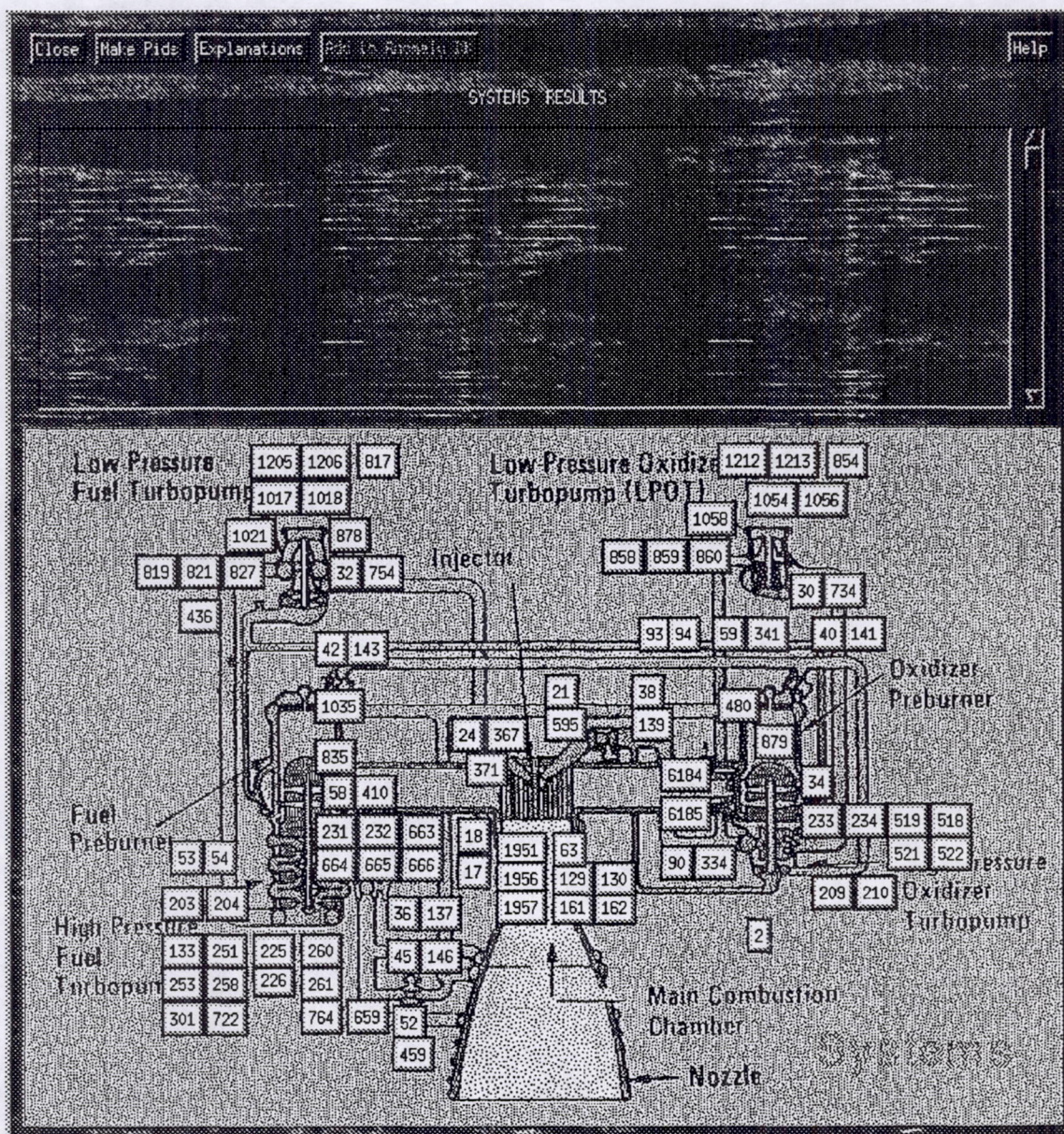


Figure 8. Systems window.

The "Create PIDs" window contains the following controls:

- Buttons:** Close, Next, First, New, Print, Reset, Update, Refresh, Help.
- Pid name:** 40
- Save** button
- x:** 221 **y:** 10
- Filename:** [empty text field]

Figure 9. "Make Pids" window for modifying PID buttons on plant diagrams.

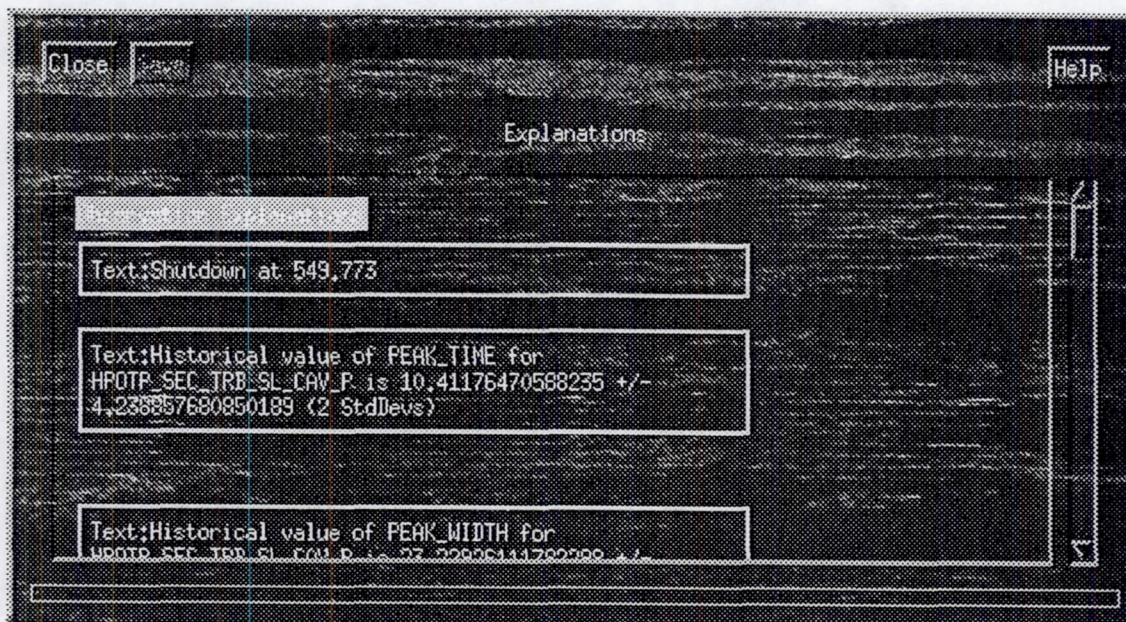


Figure 10. Explanations window for anomalies.

Plots are displayed in a window such as the one shown in Figure 11. The buttons along the top allow you to change the vertical or horizontal scales (range and time interval) of the display, or select whether full-sample or one-second-averaged data is displayed. Note that none of these options take effect until you left-click on the Replot button.

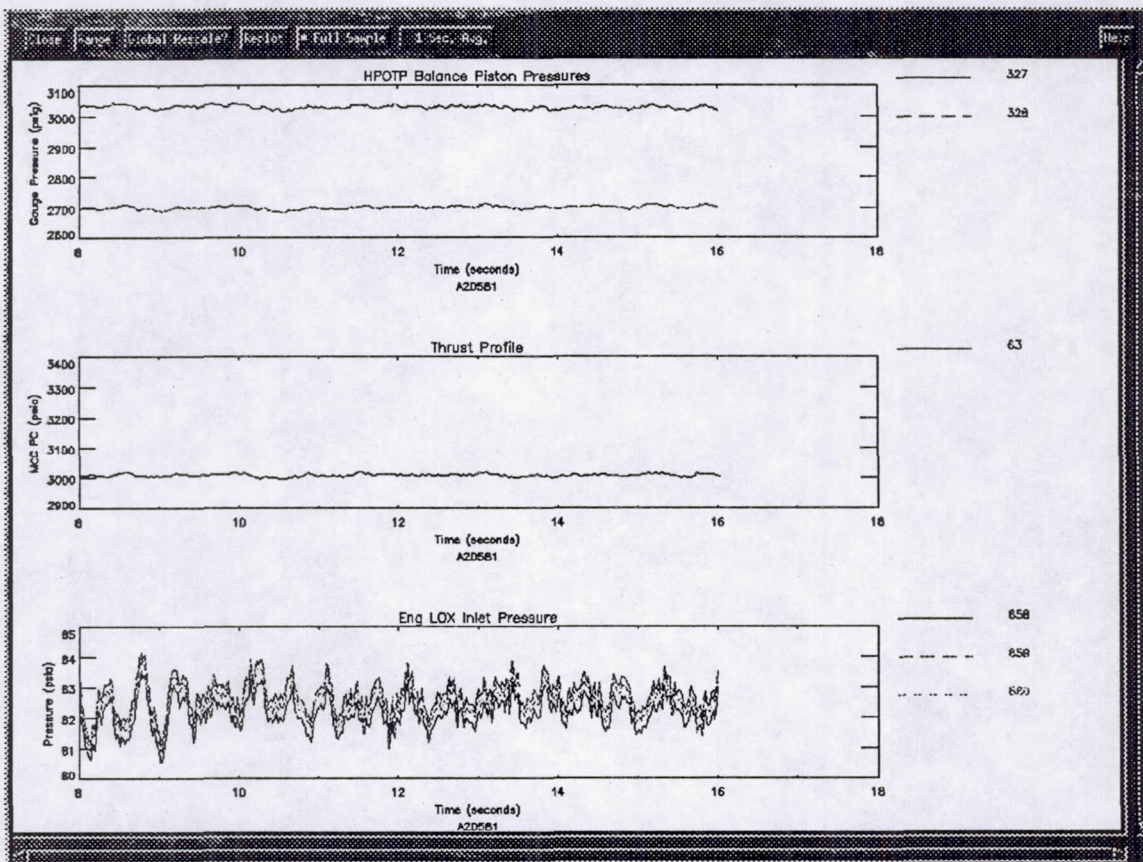


Figure 11. Plot panel for program ehms.

Anomaly Database Update and Review

The Anomaly Database gives analysts a mechanism for tracking engine performance troubles. The following tasks can be performed with the Anomaly Database:

- after a test and data review, log and categorize any anomalies in SSME test data, along with expert assessments relating to the anomalies, actions taken, and the corroborating sensor data, if desired
- retrieve data describing previously observed anomalies for analyzing patterns in engine performance
- retrieve all examples of classes of anomalies along with experts' determination of their causes for the purpose of training new analysts.

To start execution of the Anomaly Database, type `anom` at the Unix command line. After several minutes and a number of messages, the window shown in Figure 12 will be displayed. This window is the request screen used to enter the facts about the anomalies to be retrieved. The "Go" button on the menu bar is used to submit a request to the database to return all of the anomalies that match the fields filled in on the request screen. The "Options" button provides access to a short pull-down menu with the options "Clear" and "Quit". The "Clear" option clears the window of all user-entered material, useful when a user has finished looking at one record and is ready to request another, and the "Quit" option exits the Anomaly Database.

Options		Go		Help	
Status: Ready for a new query.					
Edit Mode Read		Fixed Fields: Test Number: Test Date: Engine Number:		Anomaly Title Location: BLANK	
				Type: Problem: Sensor Type:	
Misc. Fields Power Level: Test Phase: BLANK		Spec Violation Spec Violation: BLANK		Free Form Text 	
Engine Fit/Dev: BLANK LFU Fit/Dev: BLANK					
Anomaly Title 		Data Stored 		User Info Logged By: Logged Date:	

Figure 12. Main panel for the program anom.

The "Edit Mode" menu button indicates the type of database access that is being requested. For most users, this should always be set to "Read". The remaining options ("Add", "Delete", and "Modify") are used for adding new anomalies and deleting or modifying existing ones and are limited to usage by users with write permission for the database.

The remaining areas of the window are used to place restrictions on the list of anomalies to be retrieved when the "Go" button is pressed. If no data is entered into these areas, the request would return all anomalies in the database, potentially a long list. Otherwise, all anomalies which match the information entered in the window will be retrieved when the "Go" button is pressed. In general, if the user wishes to retrieve one particular anomaly, the simplest way to find it, without retrieving a large number of other anomalies that must be browsed through as well, is to fill in the fields that will place the most severe restriction on the list retrieved.

The "Fixed Fields" area includes three text fields: "Test Number", "Test Date", and "Engine Number". The test number must be entered as a six character string such as A20531 if used. The test date, if used, must be entered as an eight character string composed of two month digits, a slash, two day digits, another slash, and two year digits, e.g., 02/12/90. A question mark, "?", may be used as a wildcard for a digit anywhere in the date field. To retrieve anomalies by matching on the engine number, the engine number must be entered as a four digit number, including leading zeros, e.g., 0213.

The "Anomaly Title" area is used to restrict the types of anomalies to be retrieved. The "Location" menu button allows the user to specify whether the anomaly was in a particular LRU (option "LRU"), a sensor (option "Sensor"), or a system problem (option "System"). The default option "BLANK" means no restrictions will be made on the anomalies retrieved, based on this field. Selecting another option, the query window will change to allow the user to specify more about the anomaly. For example, if "LRU" is selected, the "LRU Unit #" field will be added to the "Fixed Fields" area of the query window and the button menu for "Type" will be displayed in the "Anomaly Title" area. The changed areas for the "LRU" and "System" options are shown in Figures 13 and 14, respectively. The options on the other menu buttons in the "Anomaly Title" area, when displayed, will depend on the selections already made at the time the menu button is displayed. For example, if "LRU" was selected for the location, the options on the "Type" menu button will be a list of LRUs, or if "Sensor" was selected for the location, the "Type" menu button will be used to specify which LRU the sensor was monitoring.

The screenshot shows a window divided into two main sections. The left section, titled "Fixed Fields", contains four input fields: "Test Number:", "Test Date:", "Engine Number:", and "LRU Unit #:". The right section, titled "Anomaly Title", contains four menu buttons: "Location:" (with "LRU" selected), "Type:" (with "BLANK" selected), "Problem:", and "Sensor Type:".

Figure 13. "Fixed Fields" and "Anomaly Title" areas of the Anomaly Database query window after selecting "LRU" from the "Location" menu button.

The screenshot shows a window titled "Anomaly Title" with four menu buttons: "Location:" (with "System" selected), "Type:" (with "BLANK" selected), "Problem:", and "Sensor Type:".

Figure 14. "Anomaly Title" area of the Anomaly Database query window after selecting "System" from the "Location" menu button.

The "Misc. Fields" area of the query window provides for other miscellaneous restrictions on the anomalies to be retrieved. The "Test Phase" menu button allows the selection of a test phase (prestart, mainstage, etc.), the "Engine Flt/Dev" menu button allows restricting queries to flight engines or development engines, and the "LRU Flt/Dev" menu button allows restricting queries to flight LRUs or development LRUs.

The "Spec Violation" menu button in the "Spec Violation" area may be used to specify a particular type of Spec Violation, such as Greenrun, ICD, ICC, Max Qual, etc.

The "Logged By" and "Logged Date" fields of the "User Info" area are used for the user ID of the person entering the records and the date on which the anomaly was entered into the database. These search fields are mainly for the convenience of users who are editing records in the database.

The remaining areas, "Free Form Text", "Anomaly Time", and "Data Stored" are not used in preparing a query for a search of the anomaly database. They are filled in by the program when a query has been completed or are used by the other "Edit Mode" options.

After a query is processed, the "Status" line will indicate the number of anomalies that satisfied the query. If only one anomaly satisfied the query, the data for that anomaly will be displayed on the main panel. If the query was satisfied by more than one anomaly, the selection window shown in Figure 15 will also be displayed with a list of anomalies which satisfy the query. A particular anomaly can then be selected from the scrolling list in the center of the window. When the "Load" button is pressed, this window will close and the data for the selected anomaly will be displayed on the main panel. Figure 16 shows the main panel after a single anomaly has been selected.

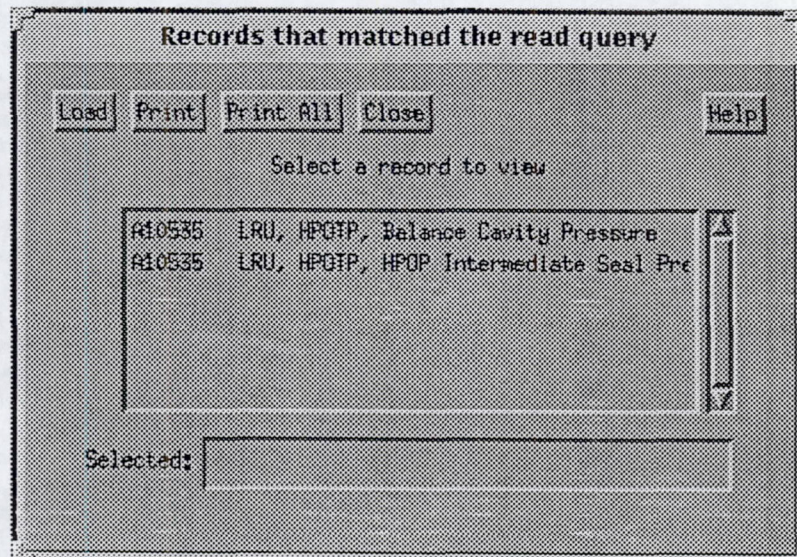


Figure 15. Selection panel for anom displayed when more than one anomaly satisfies a query or the "Read Selection" button on the main panel is pressed.

If there was more than one anomaly that satisfied the query, the "Read Selection" button in the "Edit Mode" area of the main panel can be used to redisplay the selection window for selecting other anomalies to display. The "Print" button in the selection window is used to print the data for the selected anomaly and the "Print All" button is used to print the data for all anomalies that satisfied the query. If there was only one anomaly that satisfied a query, the "Read Selection" button is used to display the selection window to provide access to the print buttons.

Options		Go		Help	
Status: The selected record has been loaded for viewing.					
Edit Mode Read Read Selection		Fixed Fields Test Number: H10535 Test Date: 07/18/07 Engine Number: 2105 LRU Unit #: 8809		Anomaly Title Location: LRU Type: HFUIP Problem: Balance Cavity Pressure Sensor Type:	
Misc. Fields Power Level: 109 Test Phase: Mainstage Engine Fit/Dev: DEV LRU Fit/Dev: DEV		Spec Violation Spec Violation: NO		Free Form Text Assessment Analysis Results Actions Taken	
Anomaly Time Start Time (sec): 0.00 Duration (sec): 0.00		Data Stored Data Stored: YES PIDs and Data		User Info Logged By: Logged Date: 00/00/00	

Figure 16. Main panel for the program anom after a data retrieval.

The three buttons in the "Free Form Text" area ("Assessment", "Analysis Results", and "Actions Taken") are used to pop up text windows containing additional textual material that is stored as part of the anomaly record. If the "Data Stored" button in the "Data Stored" area says "YES", clicking on the "PIDs and Data" button will display an auxiliary window used for displaying the data graphically.

If any of the data filled in by the user in preparing a query does not match the expected format for that field, a dialog box will be displayed containing a suggestion for editing the field when the "Go" button is pressed and the query will not be submitted to the database. The user should then close the dialog box and modify the contents of the indicated field before attempting to resubmit the query.

The "Add" option from the "Edit Mode" menu button is used for adding new anomalies to the database. Figure 17 shows the main panel with the "Add" option selected before any data for the anomaly has been entered.

Options		Go		Help	
Status: Ready for a new record to be entered.					
Edit Mode Add		Fixed Fields Test Number: Test Date: Engine Number: LRU Unit #:		Anomaly Title Location: LRU Type: Combustion Devices Problem: HCL Fuel Leak Sensor Type:	
Misc. Fields Power Level: Test Phase: Mainstage Engine Flt/Dew: FLT LRU Flt/Dew: FLT		Spec Violation Spec Violation: NO		Free Form Text Assessment Analysis Results Actions Taken	
Anomaly Time Start Time (sec): Duration (sec):		Data Stored Data Stored: NO		User Info Logged By: Logged Date: 01/31/95	

Figure 17. Main panel for the program anom when using the "Add" command.

III. Maintaining the Historical Database

The HPOTP diagnostic system utilizes a historical database of parameters for use in statistical analyses. This database is currently stored in the TekBase database named SSME_DB in the table named HISTORY (for Rocketdyne pumps) and ATDHSTRY (for Pratt & Whitney pumps). Each row in these tables contains information about a single parameter value for a single test, and has the following four columns:

TESTID	The test ID.
PARAM	The name of the parameter (e.g., HPOTP_PRI_TRB_SL_DR_P).
TYPE	The type of the parameter (e.g., PEAK_WIDTH).
VALUE	The value of the parameter (e.g., 3.27).
OK_TO_USE	A Boolean (TRUE or FALSE) value which indicates if this value should be used for future statistical analyses.

In addition, the ADTHSTRY table also contains a Boolean column IS_HOT which is used to classify the pump as having either a hot or cold "ski slope". This effectively defines three classes of pumps for which statistics are gathered: Rocketdyne, "Hot" ATD, and "Cold" ATD.

By default, OK_TO_USE values are always TRUE. However, if a HPOTP experiences a significant anomaly and you do not want some or all of its parameters used in future statistical analyses, simply set the appropriate OK_TO_USE values to FALSE (via Kingfisher).

Updating the Historical Database

A utility program is available which will update the historical database with parameters from a test without performing a full diagnostic analysis. To run this program, enter HPOTP_update testID at the Unix command line, where testID is a six-character string such as A20551 or A40123.

Viewing the Historical Database

A utility program is available which will provide a quick print out of the contents of the historical database. To run it, enter HPOTP_history at the UNIX command line. The program will ask you for the name of a log file to store the data in. You can either enter a valid filename, or simply a carriage return to indicate that you do not want a log file created. The program will then ask whether you want statistics for Rocketdyne, "Hot" ATD, or "Cold" ATD pumps. A printout similar to the following will be output:

```
----- A20571 (ENABLED) -----
PEAK_HEIGHT      HPOTP_SEC_TRB_SL_CAV_P    24.31
PEAK_WIDTH        HPOTP_SEC_TRB_SL_CAV_P    19.67
PEAK_TIME         HPOTP_SEC_TRB_SL_CAV_P     9.00
PEAK_HEIGHT      HPOTP_PRI_TRB_SL_DR_P    36.97
PEAK_WIDTH        HPOTP_PRI_TRB_SL_DR_P    22.66
PEAK_TIME         HPOTP_PRI_TRB_SL_DR_P     8.00
EQ_VAL            HPOTP_SEC_TRB_SL_CAV_P    11.53
EQ_VAL            HPOTP_PRI_TRB_SL_DR_P     7.75
START_VAL         HPOTP_INT_SL_PRG_P     188.91
5_TO_CUT          HPOTP_PRI_PMP_SL_DR_P     0.25
MAX_AFTER_EQ      HPOTP_PRI_PMP_SL_DR_T    427.52
104_MIN_NPS       HPOTP_BAL_CAV_P_A    3141.81
```

104_MIN_NPSP	HPOTP_BAL_CAV_P_B	3012.06
109_MAX_NPSP	HPOTP_BAL_CAV_P_A	3380.77
109_MAX_NPSP	HPOTP_BAL_CAV_P_B	3258.33
104_NOM_NPSP	HPOTP_BAL_CAV_P_A	3187.00
104_NOM_NPSP	HPOTP_BAL_CAV_P_B	3048.27

...

...

...

***** SUMMARY OF ENABLED TESTS *****

TYPE	PARAMETER	MEAN	STDDEV	N
109_MAX_NPSP	HPOTP_BAL_CAV_P_A	3286.86	77.40	11
109_MAX_NPSP	HPOTP_BAL_CAV_P_B	3076.68	178.48	11
104_MIN_NPSP	HPOTP_BAL_CAV_P_A	3053.03	75.97	22
104_MIN_NPSP	HPOTP_BAL_CAV_P_B	2863.38	143.00	22
PEAK_HEIGHT	HPOTP_SEC_TRB_SL_CAV_P	24.15	2.48	32
PEAK_WIDTH	HPOTP_SEC_TRB_SL_CAV_P	23.32	4.14	32
PEAK_TIME	HPOTP_SEC_TRB_SL_CAV_P	10.41	2.24	32
EQ_VAL	HPOTP_SEC_TRB_SL_CAV_P	12.64	1.21	34
EQ_VAL	HPOTP_PRI_TRB_SL_DR_P	8.99	1.21	34
MAX_AFTER_EQ	HPOTP_PRI_PMP_SL_DR_T	416.55	20.73	34
PEAK_HEIGHT	HPOTP_PRI_TRB_SL_DR_P	33.47	2.91	36
PEAK_WIDTH	HPOTP_PRI_TRB_SL_DR_P	25.14	5.41	36
PEAK_TIME	HPOTP_PRI_TRB_SL_DR_P	8.83	1.32	36
104_NOM_NPSP	HPOTP_BAL_CAV_P_A	3100.58	76.04	37
104_NOM_NPSP	HPOTP_BAL_CAV_P_B	2899.31	143.76	37
5_TO_CUT	HPOTP_PRI_PMP_SL_DR_P	-0.02	0.23	37
START_VAL	HPOTP_INT_SL_PRG_P	190.92	4.29	38

SSME Post-Test Diagnostic System Systems Section

**Final Report
Attachment #2**

Programmer's Guide

Post-Test

Diagnostic System

(PTDS)

Programmer's Guide

Prepared by:
J. Allen Crider
Computer Sciences Corporation
22 February, 1995

Table of Contents

Table of Contents	i
Acknowledgments	iii
Introduction	1
1. Execution of Individual PTDS Programs	1
1.1 Session Manager (smgr)	1
1.2 Feature Extractor (features)	1
1.3 External Effects (external)	2
2. Database Tables	4
2.1 Space Shuttle Main Engine Database (SSME_DB)	4
2.2 Session Manager Database (SESS_MGR)	28
2.3 External Effects Database (EXTERNAL)	29
3. New Data (new_data)	30
3.1. Source Files	30
3.2 Header Files	30
3.3 Functions	30
3.4 Cflow output	30
4. Session Manager (smgr)	31
4.1 Source Files	31
4.2 Header File	32
4.3 Defined Constants	32
4.4 Defined Types	32
4.5 Global Variables	33
4.6 Functions	33
4.7 Cflow output	36
5. Features (features)	38
5.1 Source Files	38
5.2 Header Files	39
5.3 Defined Constants	39
5.4 Defined Types	42
5.5 Global Variables	46
5.6 Functions	46
5.7 Algorithm Comments	68
5.7.1 General Comments	68
5.7.2 DifferentThan Module	68
5.8 Cflow output	70
6. External Effects Program (external)	74

6.1 Source Files.....	74
6.2 Header Files.....	75
6.3 Defined Constants.....	75
6.4 Defined Types.....	75
6.5 Functions.....	80
6.6 Algorithm.....	87
6.6.1 Notation.....	87
6.6.2 Computations.....	88
6.7 Cflow output.....	89
7. HPOTP Module.....	90
7.1 Modules.....	90
7.1.1 Executive.....	90
7.1.2 Feature Extraction.....	91
7.1.3 HPOTP Sensor Validation.....	91
7.1.4 Redundancy Management.....	92
7.1.5 Statistics Module.....	92
7.1.6 Anomaly Detection & Diagnosis.....	92
7.1.7 Green Run Specifications Check.....	93
7.1.8 Supporting Plot Generation.....	93
7.1.9 Output of Results.....	93
7.2 Anomalies Currently Detected by the HPOTP Diagnostic System.....	93
7.2.1 General Anomalies.....	93
7.2.2 Green Run Specifications.....	99
8. Common Function Library for PTDS.....	100
8.1 Source Files.....	100
8.2 Header Files.....	100
8.3 Defined Constants.....	101
8.4 Defined Types.....	102
8.5 Global Variables.....	105
8.6 Functions.....	106

Acknowledgments

This User's Guide and Programmer's Guide contains contributions from various members of the Post-Test Diagnostic System (PTDS) development team. The members of the development team are

Rick Ballard

Tim Bickmore

J. Allen Crider

Chris Fulton

Bill Maul

Catherine McLeod

Claudia Meyer

Virginia Tickle

Luis Trevino

June Zakrajsek

Introduction

This portion of the manual provides information useful to the maintainers of the Post-Test Diagnostic System (PTDS) and developers of new modules for the system. The first section contains information on running individual programs that are normally not executed directly by the average user. The following section includes descriptions of all of the TekBase databases utilized by PTDS. The remaining sections include descriptions of all source files, header files, and functions for each module, program, or library included in PTDS. In addition, a partial listing of the output from the Unix utility `cflow` is provided for each program. All references to PTDS functions and Metrica library functions are included in these listings, but references to standard library functions and X Windows library functions have been deleted.

1. Execution of Individual PTDS Programs

This section provides usage information on programs included in PTDS which are not normally executed by the average user. All of these programs are run automatically as a result of running `new_data` and should only be run standalone by developers and maintainers of the system.

1.1 Session Manager (`smgr`)

The session manager, `smgr`, runs all tests queued by the program `new_data` through a predetermined set of PTDS modules. It is started automatically by `new_data` whenever a new test is queued if it is not already running. It can also be started from the command line with no arguments, although this is not normally done.

1.2 Feature Extractor (`features`)

The feature extractor program, `features`, provides the basic information on engine behavior necessary for operation of the expert modules. The expert modules use the sensor trace "features" reported by the feature extractor to reason about the health of an SSME component or assembly.

The feature extractor is currently capable of detecting the following generalized sensor trace features:

- **peaks** (All peaks or only the primary peak, where primary peak is defined as the peak having the greatest magnitude on the interval of interest)
- **spikes**
- **erratic behavior**
- **level shifts**
- **redline violations**

Sensor traces may also be statistically compared to determine the likelihood that they represent the "same" (two samples of data from the same parent distribution) or a different measurement or differ by a constant offset. This capability is provided by the feature extractor `DifferentThan`. In addition to detecting the general features described above, the feature extractor is also capable of detecting more specific behaviors of the SSME such as changes in the net force exerted on the balance piston, and preburner pump bistability.

When invoked, the feature extractor reads a general command table, F_COMM in the SSME_DB database, that provides the program with basic information such as to what type of features to look for and in what measurements (PIDs) to look for them. More specific information, such as the start and stop times of the search, and in the case of peaks, what type of model to fit to, are determined by the program at run time. By writing a set of general commands to the feature extractor, which are valid for all tests, consistent behavior is assured. In light of the consistent manner in which features are collected, the results may be used to accurately monitor the health of an SSME component.

Feature extraction is initiated by the session manager after the session manager has been notified of the arrival of new data. It can also be started from the command line with a test ID as the only argument. Both controller and facilities data for a test must be loaded prior to feature extraction. The first operation performed by the feature extractor is an analysis of the thrust profile for the test of interest. Periods of constant thrust are detected and classified by start time, stop time and percent thrust. This information is then used to provide parameters for each feature extraction module. Only features occurring (?) during times of constant thrust are extracted. This provides protection against expected transients in the data which could manifest themselves as interesting features.

The feature extractor is designed to be run once for each analysis of an SSME test. A general command table, as referred to above, has been provided for the extraction of those features necessary for HPOTP analysis. No changes to this command table, or test specific setup is required. To extract features for the analysis of another module, appropriate commands must be appended to the command table. To alter any aspect of the program or command table between tests, aside from additions to the command table pertaining to additional modules, would make any further comparisons between tests invalid. The program needs only to be called with a different test ID on the command line to provide features for another test.

1.3 External Effects (external)

The External Effects program, external, removes the effects of several independent PIDs from the differences of dependent PIDs for two tests, resulting in normalized delta PIDs for the dependent parameters. The independent PIDs for which effects are removed are power level, mixture ratio, low pressure fuel pump (LPFP) inlet pressure, low pressure Ox pump (LPOP) inlet pressure, LPFP inlet temperature, and LPOP inlet temperature. In addition, the module uses the results from the Hardware Change Reporter to adjust the normalized delta PIDs for changes in the hardware used for the two tests. The resulting data can then be examined for anomalies.

External effects is automatically run as part of the Systems module. The typical user will normally not need to run this program separately. The remainder of this section describes what a user needs to know to run external effects separately.

The external effects module depends on several TekBase database tables containing some necessary information required by the program in order to get complete results. The GAINS table in the EXTERNAL database is used to determine the dependent PIDs and the values required in computing the effects of the independent PIDs. Further discussion of the EXTERNAL database may be found in the Database Tables section below and in the algorithm description for this program below. Other database tables referenced are in the SSME_DB database. The CMP_DESC table is referenced to determine the comparison test to be used and the TST_HW table is used to determine the HPOT pump type used during the current test. If the required data is not available from these tables, the program will output an error message and terminate. The RED_S_C table is used to determine a valid PID to be used for each independent PID except mixture ratio and for each dependent PID. If any independent PIDs are missing for either the current test or comparison test, the program will output an error message and terminate. If

any dependent PIDs are missing for either test, no data for that PID will be included in the output file generated by the program. This table is normally populated by the Sensor Validation module of PTDS. The DELTAS table is used to determine the constants to be added to the normalized delta PIDs to account for changes in the hardware. If there is no entry for a particular dependent PID for the current test, no adjustment is made to the normalized delta PID for that parameter. This table is normally populated by the Hardware Change Reporter module of PTDS.

Two environment variables are checked by the program. The variable QYHOST is checked to determine the machine to use as the TekBase server for database queries. If the environment variable is not set, the program uses the default machine, jetson. The file location library used to determine the directories containing data files uses the environment variable FL_RC_PATH to determine the location of the resource file .fllibrc. (See the File Location library documentation for more information.)

The command line for running the external effects module is

```
external CurrentTest [normalizeFlg]
```

where

CurrentTest is the identifier for the current test. This may be either the name of a single test data file or the name of a database directory file which contains a list of the data files containing the test data. Currently, the program assumes that if the identifier contains a period, it is the name of a single test data file; otherwise, it is assumed to be the name of a database directory file. The program uses the File Location library and the file .fllibrc to determine the locations of any database directory files and test data files. The CMP_DESC table from the SSME_DB database is searched to determine the appropriate comparison test corresponding to *CurrentTest* to be used by the program.

normalizeFlg is an optional flag for indicating whether the user wishes to calculate normalized delta PIDs (i.e., delta PIDs with the external effects removed) or only delta PIDs (i.e., PIDs which are the result of subtracting the PIDs for the comparison test from the corresponding PIDs for the current test with no adjustment made for external effects). If the first character of *normalizeFlg* is D (must be upper case), then only delta PIDs are computed and written to the output file. If *normalizeFlg* is omitted or if the first character of *normalizeFlg* is anything other than D, then the external effects are removed after calculating the delta PIDs and only the normalized PIDs are written to the output file.

If no arguments or more than two arguments are provided on the command line, the program prints the message

```
Usage: external CurrentTest [D]
      D - produce unnormalized deltas
```

and exits. The program will also print an appropriate error message and exit when some error conditions are detected, such as a required test data file or database directory file missing or required data missing from a referenced database table. A warning message will be printed if a test data file listed in a required database directory file is missing, but the program will continue to execute if all of the independent PIDs can be found in the existing test data files. A warning message is also printed if no entry can be found in the RED_S_C table in the SSME_DB database for a dependent PID description for the current test or for the comparison test. If no errors are encountered, the results are written to a binary data file in SSME format. If the first character of *normalizeFlg* is D, the name of the output file will be *CurrentTest.DPID*. Otherwise, the name of the output file will be *CurrentTest.NPID*. In addition, the following items are written to stdout: the cutoff time for the current test, a list of the

dependent PIDs for which deltas were computed, and a message including the name of the binary output file.

Example: When external is run with the command line

```
external a20584 D
```

the following information is written to stdout:

```
Comparison test is a20583
cutoff = 299.869
      86      52      17      835      209      90      59      58      480      24      15
659      18      93      21      231      233      32      260      30      2      142
140 1205 1212
Data is in file a20584.DPID
```

2. Database Tables

2.1 Space Shuttle Main Engine Database (SSME_DB)

Table ANOMDATA

ANOM#	
PID#	
REC#	
V1	
V2	
V3	
V4	
V5	
V6	
V7	
V8	
V9	
V10	
V11	
V12	
V13	
V14	
V15	
V16	
V17	

V18	
V19	
V20	
V21	
V22	
V23	
V24	
V25	
V26	
V28	
V29	
V30	
V31	
V32	
V33	
V34	
V35	
V36	
V38	
V39	
V40	
V41	
V42	
V43	
V44	
V45	
V46	
V48	
V49	
V50	

Table ANOMDATO

ANOM#	
-------	--

PID#	
DESC	
RATE	
START_TIME	
END_TIME	

Table ANOMINFO

ANOM#	Unique number given to the anomaly
TEST#	Test ID for test where anomaly occurred.
MONTH, DAY, YEAR	Month, day, and year that the test was run.
TEST_PHASE	
POWER_LEVEL	
ANOM_ST_TIME	Time that the anomaly started.
ANOM_DUR	Duration of the anomaly.
ENGINE#	
ANOM_LOC	
ANOM_TYPE	
ANOM_PROBLEM	
SENSOR_TYPE	
LRU_UNIT#	
FL_OR_DEV_EG	
FL_OR_DV_LRU	
SPEC_VIOLAT	
SPEC_VIOLCR	
ANOM_ANALYS	
ANOM_ACTION	
ANOM_ASSESS	
START_TIME	
END_TIME	
USER_NAME	User that entered data into the database.
USER_MONTH, USER_DAY, USER_YEAR	Month, day, and year that user entered data into the database.

Table ANOMPRDE

ANOM_LOC	
ANOM_TYPE	
ANOM_PROBLEM	

Table ANOMTEXT

ANOM#	Unique anomaly number
COMMENT#	ID number specifying comment as action, analysis, or resolution.
COMMENT	Text of comment.

Table ANOMTPH

PHASE	
-------	--

Table ANOM_SPV

VIOLATION	Available anomaly.
-----------	--------------------

Table ANOM_SST

SENSOR	Available sensor types.
--------	-------------------------

The ATD history table, ATDHSTRY, contains the historical database of parameters for Pratt & Whitney (ATD) pumps utilized by the HPOTP diagnostic system. It is updated automatically by the HPOTP module or by the program HPOTP_update. (See Section III of the User's Guide for more information.)

Table ATDHSTRY

TESTID	The test ID.
TYPE	The type of the parameter.
PARAM	The name of the parameter.
VALUE	The value of the parameter.
IS_HOT	Classifies the pump as having either a hot or cold "ski slope".
OK_TO_USE	Whether the parameter is to be used in future statistical analyses.

Table CFE_LIMS

MODULENAME	
PID	
REDLINE_NAME	

REDLINE	
---------	--

The comparison description table, CMP_DESC, is used to store information about the comparison test to be used for each test. It is normally updated by new_data, but it may be updated manually. All columns are used by the Hardware Change Reporter. The first two columns are also used by the Case Based Reasoner and external.

Table CMP_DESC

TEST_ID	Test ID for a current test.
COMP_TEST_ID	Test ID for the test to be used as a comparison test for the current test.
CMP_START	
CMP_STOP	
DURATION	
CMP_SHUTDOWN	
POWER_LEVEL	

The table DELTAS is populated by the Hardware Change Reporter, Comparator, and Sensor Validation. The table is accessed by external during the process of modifying PIDs to account for hardware changes.

Table DELTAS

TEST_ID	Test ID for the current test.
COMP_TEST_ID	Test ID for the comparison test used.
PRODUCER	Systems sub-module which produced the delta entry.
COMPONENT	
PARAMETER	
START_TIME	
END_TIME	
START_DELTA	
END_DELTA	
UNITS	

The table D_BOOK is used by the Hardware Change Reporter and Case Based Reasoner. It is updated manually as required.

Table D_BOOK

ENG_CHANGE	
MAGNITUDE	

UNITS	
TYPE	
PHASE	
ENG_FL_IN_PR	
ENG_FL_IN_T	
LPFP_SP	
HPFP_IN_PR	
HPFP_IN_T	
ENG_V_FU_FL	
HPFP_SP	
HPFP_DS_PR	
HPFP_DS_T	
MCC_CL_DS_PR	
MCC_CL_DS_T	
LPFT_IN_PR	
ENG_O_IN_PR	
ENG_O_IN_T	
LPOP_SP	
HPOP_IN_PR	
HPOP_SP	
HPOP_DS_PR	
MCC_O_INJ_PR	
MCC_O_INJ_T	
PBP_DS_PR	
PBP_DS_T	
MFV_POS	
MOV_POS	
CCV_POS	
OPOV_POS	
FPOV_POS	
FL_PR_INT_PR	
FU_PR_INT_T	
HEX_INT_PR	

HEX_INT_T	
FPB_PC	
HPFT_DS_T	
OPB_PC	
HPOT_DS_T	
MCC_HG_IN_PR	
VOL_LOX_FL	
VOL_FUEL_FL	
CBR	

Table EXPECT_1

TEST_ID	Test ID for the current test.
MODULE	
NUMBER	
TYPE	
PID	
START_TIME	
STOP_TIME	
SIGN	
CHG_MAG	

The table EXPLANE is populated by the Hardware Change Reporter, Case Based Reasoner, and Sensor Validation.

Table EXPLANE

TEST_ID	
MODULE	
NUMBER	
TYPE	
DEGREE	
START_TIME	
STOP_TIME	
DESCRIPTION	
COMP_ID1	
COMP_ID2	

DATA_PARAM	
DATA_DELTA	

Table FILTBIAC

TEST_ID	
MODULE	
PID	
START_TIME	
END_TIME	
SIGN	
PID_VALUE	
TEST_TYPE	

The feature bistability table, F_BISTAB, is updated by the feature extractor whenever the FindBistable module finds any bistability features.

Table F_BISTAB

MODULE	PTDS module which requested this feature.
TESTID	Test ID for the current test.
FEAT_NUM	Unique number for this test.
SENSOR	PID where the feature was found.
FIT_START, FIT_END	Start and end times of the fit interval where this feature was found.
THRUST_LEVEL	Thrust level on the interval where this feature was found.
SENSOR_LABEL	PID description for this sensor.

The feature commands table, F_COMM, contains the commands processed by the feature extractor for each test processed by PTDS. The table must be updated manually when new modules are added to the feature extractor or new commands are required for new or existing PTDS modules.

Table F_COMM

EXPERT	This character string indicates the name of the expert module which requests the feature. The feature extractor runs only once per test so the features needed by all expert modules are extracted at the same time. This field is saved in the feature tables so that database queries may be issued for all features requested for use by a certain expert module.
--------	--

SENSOR	This is a standardized string describing the measurement to be searched for the given class of feature. This string is used to look up the appropriate PID name which is an index into the data tables. For example, PID 63 is represented by the string "MCC Combustion Pressure, Average"
SEN_POSTFIX	This indicates the use of either full sample data or one second average. The former is indicated by entering an "F" in this column while the latter is indicated by an "A". This field is used to qualify the contents of the SENSOR field which indicates which PID to operate on but does not specify whether to use sample rates of the raw data, or one-second averages computed by the PTDS.
MODULENAME	This is a string representing the feature extraction module to be called. The names of the available modules are as follows: BalancePistonCompare, DeltaLevelShift, DifferentThan, FindBistable, FindErraticBehavior, FindLevelShift, FindPeak, FindSpike, IsFlat, RedlineCheck.
STARTTIME, ENDTIME	These character fields contain strings indicating the time at which feature extraction is to start and stop for this measurement. The times can be specified as an integer value or as one of the following generic strings which represent times of interest common to all tests: bot - beginning of test data; eot - end of test data; cutoff - engine cutoff time; ts_eq - time at which turbine seal equilibrium is reached; lox_eq - time at which LOX seal equilibrium is reached. NOTE: Setting STARTTIME = ENDTIME indicates the special case where all periods of constant thrust are examined for the requested feature.
PARAM1, PARAM2, PARAM3, PARAM4, PARAM5	These character fields contain parameters specific to the named extraction module. Depending on the module, some, all or none of these fields may be used. In the event that a field is unused, its contents are irrelevant. Unused fields have been filled with an X for ease of inspection.
SENSOR_LABEL	PID description for this PID.
PARAM1_LABEL, PARAM2_LABEL, PARAM3_LABEL, PARAM4_LABEL, PARAM5_LABEL	Archaic routine specific parameters.

The feature different than table, F_DIFTHA, is updated by the feature extractor whenever the DifferentThan module or BalancePistonCompare modules find any features. The table is used by Sensor Validation.

Table F_DIFTHA

MODULE	PTDS module which requested this feature.
TESTID	Test ID for the current test.
FEAT_NUM	Unique number for this test.
SENSOR	PID where the feature was found for DifferentThan features or an indicator of the form PID#-PID# for the composite data where the feature is found for BalancePistonCompare features.
COMP_TESTID	Test ID for the comparison test.
COMP_SENSOR	Comparison test PID.
START_TIME, END_TIME	Start and end times of the fit interval where this feature was found.
CHI_SQUARE	Comparison statistic.
PROB	Comparison statistic.
COEF_W_ERR_B	Comparison statistic.
DIF_BY_OFFSE	Offset Flag.
OFFSET	Offset Flag.
OFFSET_SIGMA	Offset Flag.
THRUST_LEVEL	Thrust level on the interval where this feature was found.
SENSOR_LABEL	PID description for this PID.
COMP_SEN_LAB	PID description for the comparison PID.

The table F_DRIFT is used by Sensor Validation.

Table F_DRIFT

MODULE	PTDS module which requested this feature.
TESTID	Test ID for the current test.
FEAT_NUM	Unique number for this test.
SENSOR	PID where the feature was found.
START_TIME, END TIME	Start and stop times of the detected feature.
OFFSET	
SLOPE	Average slope of the drift.
THRUST_LEVEL	Thrust level of the detected feature.
SENSOR_LABEL	PID description for this PID.

START_MAG, END_MAG	Magnitudes of the PID at the beginning and end of the drift.
-----------------------	--

The feature erratic table, F_ERRAT, is updated by the feature extractor whenever the FindErraticBehaviour module finds any erratic behaviour features.

Table F_ERRAT

MODULE	PTDS module which requested this feature.
TESTID	Test ID for the current test.
FEAT_NUM	Unique number for this test.
SENSOR	PID where the feature was found.
START_TIME, END_TIME	Start and end times of the fit interval where this feature was found.
THRUST_LEVEL	Thrust level on the interval where this feature was found.
SENSOR_LABEL	PID description for this PID.

The feature is flat table, F_ISFLAT, is updated by the feature extractor whenever the IsFlat module finds any is flat features. The table is used by Sensor Validation.

Table F_ISFLAT

MODULE	PTDS module which requested this feature.
TESTID	Test ID for the current test.
FEAT_NUM	Unique number for this test.
SENSOR	PID where the feature was found.
START_TIME, END_TIME	Start and stop times of the detected feature.
OFFSET	
SLOPE	Average slope of the drift.
OFFSET_SIGMA	Statistic from the feature routine.
SLOPE_SIGMA	Statistic from the feature routine.
CHI_SQUARE	Statistic from the feature routine.
THRUST_LEVEL	Thrust level of the detected feature.
SENSOR_LABEL	PID description for this PID.

The feature level shift table, F_LEVSH, is updated by the feature extractor whenever the FindLevelShift or DeltaLevelShift modules find any features. The table F_LEVSH is used by the Comparator and Sensor Validation.

Table F_LEVSH

MODULE	PTDS module which requested this feature.
TESTID	Test ID for the current test.
FEAT_NUM	Unique number for this test.
SENSOR	PID where the feature was found for FindLevelShift features or an indicator of the form PID#-PID# for the composite data where the feature is found for DeltaLevelShift features.
START_TIME, END_TIME	Start and stop times of the detected feature.
LAST_MAG	Magnitude of the PID at the end of the level shift
DELTA	Size of the level shift.
THRUST_LEVEL	Thrust level of the detected feature.
SENSOR_LABEL	PID description for this PID.

The table F_NOISE is used by Sensor Validation.

Table F_NOISE

MODULE	PTDS module which requested this feature.
TESTID	Test ID for the current test.
FEAT_NUM	Unique number for this test.
SENSOR	PID where the feature was found.
START_TIME, END_TIME	Start and stop times of the detected feature.
THRUST_LEVEL	Thrust level of the detected feature.
SENSOR_LABEL	PID description for this PID.
PID	

The feature peak table, F_PEAK, is updated by the feature extractor whenever the FindPeak module finds any peak features.

Table F_PEAK

MODULE	PTDS module which requested this feature.
TESTID	Test ID for the current test.
FEAT_NUM	Unique number for this test.
SENSOR	PID where the feature was found.
PEAK_HT	Magnitude of the peak.
TAPH	Time at the maximum peak height.

FWHM	Magnitude of the peak at half height.
TAFWHM1	Time of the half magnitude on the rising slope.
TAFWHM2	Time of the half magnitude on the falling slope.
FIT_TYPE	Type of fit applied to the slope.
CHI_SQUARE	
NUM_PARAMS	
PARAM1F	
PARAM2F	
PARAM3F	
PARAM4F	
THRUST_LEVEL	Thrust level of the detected feature.
SENSOR_LABEL	PID description for this PID.
OFFSET	

The table F_RD_CC is used by Sensor Validation.

Table F_RD_CC

EXPERT	PTDS module which requested this feature.
TEST_ID	Test ID for the current test.
FEAT_NUM	Unique number for this test.
CHANNEL_A	PID one.
CHANNEL_B	PID two.
START_TIME, END_TIME	Start and stop times of the detected feature.

The feature redline violations table, F_RLVIOL, is updated by the feature extractor whenever the RedlineCheck module finds any redline violations features. The table F_RLVIOL is used by Sensor Validation.

Table F_RLVIOL

MODULE	PTDS module which requested this feature.
TESTID	Test ID for the current test.
FEAT_NUM	Unique number for this test.
SENSOR	PID where the feature was found.
PAIR_SENSOR	
VIOLAT_START, VIOLAT_END	Start and stop times of the redline violation.

CHECK_TYPE	
LIMIT_TYPE	
REDLINE	Redline value.
SENSOR_LABEL	PID description for this PID.
PR_SEN_LABEL	PID description for the paired sensor.

The feature spike table, F_ISFLAT, is updated by the feature extractor whenever the FindSpike module finds any spike features. The table F_SPIKE is used by Sensor Validation.

Table F_SPIKE

MODULE	PTDS module which requested this feature.
TESTID	Test ID for the current test.
FEAT_NUM	Unique number for this test.
SENSOR	PID where the feature was found.
START_TIME, END_TIME	Start and stop times of the detected feature.
MAGNITUDE	Magnitude of the detected spike.
THRUST_LEVEL	Thrust level of the detected feature.
SENSOR_LABEL	PID description for this PID.

The table F_THLEDE is used by the Comparator.

Table F_THLEDE

MODULE	PTDS module which requested this feature.
TESTID	Test ID for the current test.
FEAT_NUM	Unique number for this test.
SENSOR	Standard descriptor string corresponding to the thrust PID.
START_TIME, END_TIME	Start and end times of the period of constant thrust.
OFFSET, SLOPE	Parameters of a straight line fit to the data over the specified time range.
OFFSET_SIGMA, SLOPE_SIGMA	The standard deviations on the straight line fit parameters.
CHI_SQUARE	Measurement of how good the fit of a straight line to the data was.
THRUST_LEVEL	Thrust level of the detected feature, as given by OFFSET scaled to percent thrust.

The table F_ZEROSC is used by Sensor Validation.

Table F_ZEROSC

TESTID	Test ID for the current test.
MODULE	PTDS module which requested this feature.
FEAT_NUM	Unique number for this test.
AVER	Average value for the sensor.
THRUST_LEVEL	Thrust level of the detected feature.
SENSOR_LABEL	PID description for this PID.
DELTA	Difference of sensor value from expected value.
PID	PID where the feature was found.

The history table, HISTORY, contains the historical database of parameters for Rocketdyne pumps utilized by the HPOTP diagnostic system. It is updated automatically by the HPOTP module or by the program HPOTP_update. (See Section III of the User's Guide for more information.)

Table HISTORY

TESTID	The test ID.
TYPE	The type of the parameter.
PARAM	The name of the parameter.
VALUE	The value of the parameter.
OK_TO_USE	Whether the parameter is to be used in future statistical analyses.

Table KONFLICT

TEST_ID	
FMODE1	
FMODE2	
START_TIME	
STOP_TIME	

The table PARAINFO is used by the Comparator, Case Based Reasoner, and Sensor Validation. It is updated manually as required.

Table PARAINFO

SENSOR_LABEL	
UNITS	
NORM_VALUE	
SIG_CHANGE	

PID_PACKAGE	
UNIT_LABEL	

Table PHASES

TEST_ID	Test ID for the current test.
MODULE	PTDS module initially generating this feature.
PHASE	Phase name.
START_TIME, STOP_NAME	Start and stop times for the phase.

Table PIDINFO

TEST_ID	Test ID for the current test.
PID	PID name.
UNITS	PID units.
DESCR	PID description as in flat file.
STD_DESCR	Standard PID description.
RATE	Sample rate.
START_TIME, END_TIME	Start and stop times of the PID as in flat file.
SENSOR_LABEL	PID description for this PID.

The table PIDS_MIA is used by Sensor Validation.

Table PIDS_MIA

TEST_ID	Test ID for the current test.
PID	Required PID that is missing from the flat file.

Table PID_DEF

PID	PID name.
DESCR	PID description as sometimes read in the flat file.
STD_DESCR	Standardized PID description.
SENSOR_LABEL	PID description for this PID.

Table PID_DEF2

PID	
DESCR	
STD_DESCR	

SENSOR_LABEL	
--------------	--

Table PID_ORID

TEST_ID	Test ID for the current test.
PID	Nominally standard PID name.
OVERRIDE_PID	

The table PLOTINFO is populated by the Case Based Reasoner and Sensor Validation.

Table PLOTINFO

NAME	
POST_NUMBER	Unique anomaly number.
PLOT_TYPE	
MODULE	Module submitting
CUR_TESTID	Test ID for the current test.
PREV_TESTID	Test ID for the previous test if applicable.
NUM_PLOTS	Number of plots (maximum 10).
FULL_SAMPLE1, FULL_SAMPLE2, FULL_SAMPLE3	Flags specifying whether to plot with full sample or 1-second averages for plots 1, 2, and 3, respectively.
NUM_CURVES1, NUM_CURVES2, NUM_CURVES3	Number of curves for each of plots 1, 2, and 3, respectively; sum must be equal to NUM_PLOTS.
START_TIME1, START_TIME2, START_TIME3	Start times for plots 1, 2, and 3, respectively.
END_TIME1, END_TIME2, END_TIME3	End times for plots 1, 2, and 3, respectively.
TITLE1, TITLE2, TITLE3	Titles for plots 1, 2, and 3, respectively.
SUBTITLE1, SUBTITLE2, SUBTITLE3	Subtitles for plots 1, 2, and 3, respectively.
XTITLE1, XTITLE2, XTITLE3	Titles for the x-axes for plots 1, 2, and 3, respectively.
YTITLE1, YTITLE2, YTITLE3	Titles for the y-axes for plots 1, 2, and 3, respectively.

PID1, PID2, PID3, PID4, PID5, PID6, PID7, PID8, PID9, PID10	PIDs used for each plot, up to NUM_PLOTS.
WHICH_TEST1, WHICH_TEST2, WHICH_TEST3, WHICH_TEST4, WHICH_TEST5, WHICH_TEST6, WHICH_TEST7, WHICH_TEST8, WHICH_TEST9, WHICH_TEST10	Test ID used for each plot, up to NUM_PLOTS.
LEG_LABEL1, LEG_LABEL2, LEG_LABEL3, LEG_LABEL4, LEG_LABEL5, LEG_LABEL6, LEG_LABEL7, LEG_LABEL8, LEG_LABEL9, LEG_LABEL10	Legend for each plot, up to NUM_PLOTS.
SET_NUM	

The table POSCAUSE is populated by the Case Based Reasoner.

Table POSCAUSE

TEST_ID	
MODULE	
NUMBER	
ENG_CHANGE	
START_TIME	
STOP_TIME	
DESCRIPTION2	
SCORE	
RANK	
SCALED_VAL	
STAND_DEV	
NOT_COVERED	

The table POSHW_CH is populated by the Hardware Change Reporter.

Table POSHW_CH

TEST_ID	
COMP_TEST_ID	
COMPONENT	
TYPE_CHANGE	
PARAMETER	
START_TIME	
END_TIME	
CHANGE	

The postulates table, POSTUL, is used by the Case Based Reasoner. It is populated by the Case Based Reasoner and Sensor Validation.

Table POSTUL

NAME	Unique name for the postulate.
TEST_ID	Test ID for the current test.
MODULE	PTDS module initially generating this feature.
FMODE	
POST_NUMBER	Postulate number; unique to TYPE.
PRIORITY	Plotting priority.
START_TIME, STOP_TIME	Start and stop times for the problem.
PROBLEM	Description of the problem.
TYPE	ANOMALY, OBSERVATION, or INSTRUMENTATION.
PID	PID name if TYPE = INSTRUMENTATION.

The redundant sensor choice table, RED_S_C, is used by the Case Based Reasoner and external to determine valid PIDs to be used for a particular parameter. It is populated by Sensor Validation.

Table RED_S_C

NAME	Description of the PID.
TEST_ID	Test ID for the current test.
MODULE	
PID_PACKAGE	List of redundant/related PIDs.
PID	Name of the validated PID corresponding to the PID description.

SENSOR	
SENSOR_LABEL	

The table REL_PIDS is used by Sensor Validation. It is updated manually as required.

Table REL_PIDS

MAP_NAME	Unique MAP ID.
RPID_LIST	List of related PIDs by description.

The table RL_INFO is used by Sensor Validation. It is updated manually as required.

Table RL_INFO

PID	PID name or combination.
LIMIT_TYPE	"UPPER" or "LOWER" limits.
STARTTIME, ENDTIME	Beginning and ending times that redline is applicable.
LIMIT	Limit value.
REDLINE_TIME	Minimum time of redline.

The table SEGMENT is populated by the Case Based Reasoner.

Table SEGMENT

TEST_ID	
MODULE	
SEGMENT	
START_TIME	
STOP_TIME	
MODEL	

Table TEST

TEST_ID	
START_TIME	

The table TESTINFO is used by the Comparator and Sensor Validation to get the shutdown time for a test.

Table TESTINFO

JOB_SUB_DATE	Date test submitted to new_data.
TEST_ID	Test ID for current test.

DATEX	Date of the test from flat file.
ENGINE#	Engine number from flat file.
CPIDS	
FPIDS	
COMB_DEVICES	
CONTROLLER	
NOZZLE	
MCC	
MAIN_INJ	
POWERHEAD	
HPFTP	
HPOTP	
LPFTP	
LPOTP	
ENG_SHUTDOWN	Engine shutdown time.
PREV_TESTID	Test ID for the comparison test.

The table TST_HW is used by the Hardware Change Reporter. It is also used by external to determine which set of gains are used in calculating the effects of independent parameters. Updating of this table is described under "Hardware Configuration Data Entry" in Section II of the User's Guide.

Table TST_HW

TEST_ID	Test identifier.
ENGINE_NO	Engine number for this test.
HPOTP_U_NO	High pressure oxidizer turbopump unit number.
LPOTP_U_NO	Low pressure oxidizer turbopump unit number.
HPFTP_U_NO	High pressure fuel turbopump unit number.
LPFTP_U_NO	Low pressure fuel turbopump unit number.
HPOTP_SER_NO	High pressure oxidizer turbopump serial number.
LPOTP_SER_NO	Low pressure oxidizer turbopump serial number.
HPFTP_SER_NO	High pressure fuel turbopump serial number.
LPFTP_SER_NO	Low pressure fuel turbopump serial number.
POWERHEAD_UN	Powerhead unit number.
MAIN_INJ_UN	Main injector unit number.

MCC_U_NO	Main combustion chamber unit number.
NOZZLE_U_NO	Nozzle unit number.
CONT_UNIT_NO	Controller unit number.
POWERHEAD_TY	
CONT_TYPE	
CON_SER_NO	Controller serial number.
FL_M_S_NO	Flow meter serial number.
HPFPD_SER_NO	High pressure fuel pump duct serial number.
HPFP_DUCT_TY	High pressure fuel pump duct type (Inconel or Titanium).
HEX_ORI_DIA	Heat exchanger orifice diameter.
HEX_O_SER_NO	Heat exchanger orifice serial number.
F7_ORI_DIA	F7 orifice diameter.
F7_O_SER_NO	F7 orifice serial number.
THROAT_DIA	Main combustion chamber throat diameter.
EXIT_DIA	Nozzle exit diameter.
PLUGGED_POST	Plugged posts in injector.
MCC_CRACKS	Main combustion chamber hot wall cracks.
ENL_BLCS	Enlarged boundary layer coolant holes.
HPFTP_CONT	Contractor that built the HPFT pump used in this test.
HPOTP_CONT	Contractor that built the HPOT pump used in this test.
OPOV_SER_NO	LOX preburner oxidizer valve serial number.
FPOV_SER_NO	Fuel preburner oxidizer valve serial number.
MFV_SER_NO	Main fuel valve serial number.
MOV_SER_NO	Main oxidizer valve serial number.
CCV_SER_NO	Coolant control valve serial number.
POWH_SER_NO	Powerhead serial number.
MAINI_SER_NO	Main injector serial number.
MCC_SER_NO	Main combustion chamber serial number.
NOZ_SER_NO	Nozzle serial number.

The table TST_INFO is used by the Hardware Change Reporter. Updating of this table is described under "Hardware Configuration Data Entry" in Section II of the User's Guide.

Table TST_INFO

TEST_ID	Test identifier.
TEST_DATE	Date of this test.
PLANNED_DUR	Planned duration of this test.
SHUTDOWN	Engine shutdown time (actual duration).
ACCEPT_TEST	Is this an acceptance test?
FLT_ENGINE	Is this a flight engine?
FLT_HPFTP	Is this a flight high pressure fuel turbopump?
FLT_HPOTP	Is this a flight high pressure oxidizer turbopump?
FLT_LPFTP	Is this a flight low pressure fuel turbopump?
FLT_LPOTP	Is this a flight low pressure oxidizer turbopump?
HPFP_GR_RUN	Is this a green run high pressure fuel pump?
HPOP_GR_RUN	Is this a green run high pressure oxidizer pump?
HPOP_SC_RUN	Is this a screen run high pressure oxidizer pump?
LPFP_GR_RUN	Is this a green run low pressure fuel pump?
LPOP_GR_RUN	Is this a green run low pressure oxidizer pump?

The table TST_PERF is used by the Hardware Change Reporter. Updating of this table is described under "Hardware Configuration Data Entry" in Section II of the User's Guide.

Table TST_PERF

TEST_ID	Test identifier.
HPFP_EFF_100	High pressure fuel pump efficiency at 100% power level.
HPFP_EFF_104	High pressure fuel pump efficiency at 104% power level.
HPFP_EFF_109	High pressure fuel pump efficiency at 109% power level.
HPOP_EFF_100	High pressure oxidizer pump efficiency at 100% power level.
HPOP_EFF_104	High pressure oxidizer pump efficiency at 104% power level.
HPOP_EFF_109	High pressure oxidizer pump efficiency at 109% power level.
LPFP_EFF_100	Low pressure fuel pump efficiency at 100% power level.
LPFP_EFF_104	Low pressure fuel pump efficiency at 104% power level.
LPFP_EFF_109	Low pressure fuel pump efficiency at 109% power level.
HPF_TEM_104	High pressure fuel turbine efficiency multiplier at 104% power level.

HPF_PEM_104	High pressure fuel pump efficiency multiplier at 104% power level.
HPF_PHCM_104	High pressure fuel pump head coefficient multiplier at 104% power level.
HPF_TFPM_104	
HPO_TEM_104	High pressure oxidizer turbine efficiency multiplier at 104% power level.
HPO_PEM_104	High pressure oxidizer pump efficiency multiplier at 104% power level.
HPO_PHCM_104	High pressure oxidizer pump head coefficient multiplier at 104% power level.
HPO_TFPM_104	High pressure oxidizer turbine flow parameter multiplier at 104% power level.
LPF_TEM_104	Low pressure fuel turbine efficiency multiplier at 104% power level.
LPF_PEM_104	Low pressure fuel pump efficiency multiplier at 104% power level.
LPF_PHCM_104	Low pressure fuel pump head coefficient multiplier at 104% power level.
LPF_TFPM_104	Low pressure fuel turbine flow parameter multiplier at 104% power level.
LPO_TEM_104	Low pressure oxidizer turbine efficiency multiplier at 104% power level.
LPO_PEM_104	Low pressure oxidizer pump efficiency multiplier at 104% power level.
LPO_PHCM_104	Low pressure oxidizer pump head coefficient multiplier at 104% power level.
PBP_PEM_104	Preburner pump efficiency multiplier at 104% power level.
PBP_PHCM_104	Preburner pump head coefficient multiplier at 104% power level.

Table TST_SW

TEST_ID	
C2_100	
C2_104	
C2_109	
KF_100	
KF_104	

KF_109	
--------	--

2.2 Session Manager Database (SESS_MGR)

The Session Manager database, SESS_MGR, contains five tables used by the session manager, smgr, to determine which modules have been run on a test and the order in which modules are executed.

The job table, JOB, is used by the session manager to form the command necessary to invoke a module. It is updated automatically by the Session Manager as the modules are being executed.

Table JOB

MODULE	Name of PTDS module.
TEST_ID	Test ID for test which this module is currently running

The message table, MSG, indicates which tests have been run through each module. This table is updated automatically if the module is executed through the Session Manager. It may be updated manually if a module is executed standalone.

Table MSG

MODULE	Name of PTDS module.
TEST_ID	Test ID for test for which this module has been run.
START_TIME	Time at which module began executing this test.
END_TIME	Time at which module finished executing this test.

The resource prerequisite table, PREREQ, provides the mechanism for specifying what resources are needed in order for a given module to be invoked. It must be updated each time a new module is added to PTDS which depends on other modules being run previously or the prerequisites change for an existing module.

Table PREREQ

ID	Index value for the resource.
PRE1	Index value for first prerequisite resource.
PRE2, PRE3, PRE4, PRE5, PRE6, PRE7, PRE8, PRE9, PRE10	Index value for additional prerequisite resources as applicable. Negative values indicate unused prerequisite resources.

The resource activity board table, RSRC_BRD, is the job status blackboard used by the session manager. This table is used to track which modules have run correctly, which are currently in progress or have exited with an error, and which have not yet run. The table is updated automatically by the session manager as a test is processed and is not intended to be readable by users or other applications.

Table RSRC_BRD

TEST_ID	Test identifier.
R0, R1, R2, R3, R4, R5, R6, R7, R8, R9	Resource variables encoding 127 resources with a two bit flag for each resource. A value of 0 in a resource flag indicates that resource has not yet run for this test. A value of 1 in a resource flag indicates that the resource has run correctly for this test. A value of -1 indicates that the resource is running or has exited with an error.

The resource list table, RSRC_LST, provides a cross reference between resource indices and module names and additional information about the modules. It should be updated each time a new module is added to PTDS.

Table RSRC_LST

ID	Index value for this resource.
NAME	Name of the resource, i.e., the module name.
DISPLAYABLE	Flag to indicate whether this resource is to be displayed on the test status board in EHMS; 'Y' = do, 'N' = don't display
WHATIF	Flag to indicate which resources are available to be run under the "Whatif" module.

2.3 External Effects Database (EXTERNAL)

The EXTERNAL database contains only the table GAINS. This table is used by the External Effects program, external, in calculating the effects of the independent PIDs on the dependent PIDs. The values in this table have been estimated as discussed in the algorithm description of the External Effects program below. Normally, users will have no need to modify this table unless it is determined that the estimations are not sufficiently accurate and need refinement. Additional entries will be needed for each dependent PID for each type of HPOT pump tested.

Table GAINS

A_LOCATION	Used by software which calculated the gains; not used by PTDS.
DESCRIPTION	Description of the PID.
PID_1, PID_2, PID_3, MSID_1, MSID_2	Used by software which calculated the gains; not used by PTDS.
BASELINE	
DIM_A1	Linear gain associated with the Power Level (not used because the power level effect is assumed to be nonlinear).
DIM_A2	Linear gain associated with the Mixture Ratio.
DIM_A4	Linear gain associated with the LPFP Inlet Pressure.

DIM_A5	Linear gain associated with the LPOP Inlet Pressure.
DIM_A6	Linear gain associated with the LPFP Inlet Temperature.
DIM_A7	Linear gain associated with the LPOP Inlet Temperature.
LSQCOF_1, LSQCOF_2, LSQCOF_3, LSQCOF_4, LSQCOF_5, LSQCOF_6	Coefficients of the fifth degree polynomial approximating the power level effects, where LSQCOF_1 is the constant term and LSQCOF_6 is the coefficient of the fifth degree term.
HPOTP_CONT	Contractor which built the HPOT pump for which the values in this entry are valid.

3. New Data (new_data)

3.1. Source Files

NDATA_main.c: The main program for new_data.

NDATA_create.c

NDATA_db_t.c

NDATA_markfile.c

NDATA_utils.c

SHWER_errors.c

3.2 Header Files

NDATA_defs.h

SHWER_defs.h

3.3 Functions

3.4 Cflow output

```

1  main: void*(), <NDATA_main.c 44>
3      init_PTDS_tekbase: <>
4      SHWER_Initialize: void*(), <SHWER_errors.c 80>
9          SHWER_ShowErrorClose: void*(), <SHWER_errors.c 149>
12         SHWER_ShowWarningClose: void*(), <SHWER_errors.c 172>
17         NDATA_Initialize: void*(), <NDATA_main.c 130>
20         RSRC_GetResourceList: void*(), <RSRC_dbutils_t.c 201>
21             tbl_count: <>
25             tbl_get: <>
26             STRNG_RemoveTrailingSpaces: char*(), <STRNG_utils.c 76>
29         NDATA_CreateSSMETestList: void*(), <NDATA_db_t.c 138>
30             DBCT_SetDBSession: void*(), <DBCT_utils_t.c 132>
31             tbl_count: 21
35             tbl_get: 25
37         NDATA_CreateSMGRTestList: void*(), <NDATA_db_t.c 186>

```

```

38         DBCT_SetDBSession: 30
39         tbl_count: 21
40         tbl_get: 25
41
42     time: <>
43     localtime: <>
44     strftime: <>
45
46     NDATA_CreateManagedWidgets: void*(), <NDATA_create.c 236>
47     NDATA_CreatePidOverrideDialog: struct*(), <NDATA_create.c 257>
48     NDATA_UnmanageWidgetCB: void*(), <NDATA_main.c 416>
49     NDATA_AddToListCB: void*(), <NDATA_main.c 474>
50     NDATA_AddToListUnselected: void*(), <NDATA_utils.c 38>
51     NDATA_DeleteFromListCB: void*(), <NDATA_main.c 583>
52     NDATA_ClearListCB: void*(), <NDATA_main.c 550>
53     NDATA_DeleteAllInList: void*(), <NDATA_main.c 518>
54     NDATA_PidOverrideListCB: void*(), <NDATA_main.c 438>
55
56     NDATA_CreateMainWindow: void*(), <NDATA_create.c 40>
57     NDATA_ManageWidgetCB: void*(), <NDATA_main.c 370>
58     NDATA_ManageWidget: void*(), <NDATA_main.c 390>
59     NDATA_ClearButtonCB: void*(), <NDATA_main.c 344>
60     NDATA_ExitButtonCB: void*(), <NDATA_main.c 313>
61
62     DBCT_SetDBSession: 30
63     DBCT_DBSessionDisconnect: void*(), <DBCT_utils_t.c 91>
64     tekbase_disconnect: int(), <../../DB/tekbase.c 248>
65     query_term: <>
66
67     NDATA_GoButtonCB: void*(), <NDATA_main.c 217>
68     SHWER_ShowWarning: void*(), <SHWER_errors.c 196>
69     NDATA_IsTestAlreadyPresent: int(), <NDATA_utils.c 63>
70     NDATA_InsertTestInfo: void*(), <NDATA_db_t.c 45>
71
72     DBCT_SetDBSession: 30
73     tbl_put: <>
74     tbl_update: <>
75
76     NDATA_InsertPidOverrideInfo: void*(), <NDATA_db_t.c 85>
77     DBCT_SetDBSession: 30
78     tbl_put: 136
79
80     DBCT_SetDBSession: 30
81     RSRC_InsertResourceBoardTestId: void*(), <RSRC_dbutils_t.c 48>
82     tbl_put: 136
83
84     RSRC_FindResourceId: int(), <RSRC_rlist.c 87>
85     RSRC_UpdateResourceBoardResourceValue: void*(), <RSRC_dbutils_t.c 84>
86     tbl_get: 25
87
88     RSRC_GetBits: int(), <RSRC_rlist.c 238>
89     tbl_update: 139
90
91     NDATA_CheckTestInsert: int(), <NDATA_db_t.c 238>
92     tbl_count: 21
93
94     DBCT_SetDBSession: 30
95     IsSMGRRunning: int(), <NDATA_main.c 179>
96     execlp: <>
97
98     tbl_free_all: <>

```

4. Session Manager (smgr)

4.1 Source Files

SMGR_main.c: The main program with initialization and closing functions.

SMGR_db_t.c: The functions which retrieve data from the SESS_MGR database and which update the MSG table in the database.

SMGR_job.c: The functions which determine the PTDS modules to be run on a test and which execute the modules.

SMGR_resource.c:

4.2 Header File

SMGR_defs.h: Header file containing the constant definitions and type definitions used in **SMGR_main.c**, **SMGR_db_t.c**, **SMGR_job.c**, and **SMGR_resource.c**, and declarations of all external functions defined in those files.

4.3 Defined Constants

Boolean: Used as the type for integer variables that only take on the values **True** and **False**; value **int**; defined in **SMGR_defs.h**.

False: value 0; defined in **SMGR_defs.h**.

SMGR_DetermineNewJob: value 0; defined in **SMGR_defs.h**.

SMGR_JobPathStringLength: value 200; defined in **SMGR_defs.h**.

SMGR_MaxPathLength: value 50; defined in **SMGR_defs.h**.

SMGR_SessionManagerDB: Name of the database used to determine which modules have been run on a test and the order in which modules are executed; value **"sess_mgr"**; defined in **SMGR_defs.h**.

SMGR_StartJob: value 1; defined in **SMGR_defs.h**.

SMGR_StopSession: value 2; defined in **SMGR_defs.h**.

True: value 1; defined in **SMGR_defs.h** and **STRNG_defs.h**.

4.4 Defined Types

SMGR_Job and **SMGR_PJob** (defined in **SMGR_defs.h**)

```
typedef struct SMGR_job {
    char
        module[DBFL_ResourceNameStringLength+1],
        test_id[DBFL_TestIdStringLength+1];
} SMGR_Job, *SMGR_PJob;
```

SMGR_Message and **SMGR_PMessage** (defined in **SMGR_defs.h**)

```
typedef struct SMGR_message {
    char
        module[DBFL_ResourceNameStringLength+1],
        test_id[DBFL_TestIdStringLength+1],
        start_time[DBFL_TimeStringLength+1],
        end_time[DBFL_TimeStringLength+1];
} SMGR_Message, *SMGR_PMessage;
```

SMGR_PrerequisiteList and **SMGR_PPrerequisiteList** (defined in **SMGR_defs.h**)

```
typedef struct SMGR_prerequisiteList {
```

```

    int
    id,
    num_prerequisites,
    prerequisite[DBFL_MaxNumResourceVars];
} SMGR_PrerequisiteList, *SMGR_PPrerequisiteList;

```

4.5 Global Variables

SMGR_CurrentTestId: char [DBFL_TestIdStringLength+1]; defined in SMGR_defs.h.

SMGR_NumPrerequisites: int; defined in SMGR_defs.h.

SMGR_RSRCExeDir: char *; defined in SMGR_defs.h.

SMGR_ThePrerequisiteList: SMGR_PrerequisiteList *; defined in SMGR_defs.h.

4.6 Functions

IsAnotherSMGRRunning (declaration in SMGR_defs.h, definition in SMGR_main.c)

```
int IsAnotherSMGRRunning ()
```

This function determines whether another session manager process is already running and writes the current process ID to the lock file smgr_lock if another session manager process is not running. The return value is zero if another session manager process is running or one otherwise.

RemoveLockFile (declaration in SMGR_defs.h, definition in SMGR_main.c)

```
void RemoveLockFile ()
```

This function removes the session manager lock file smgr_lock upon completion of the tests just before exiting the session manager.

SaveSMGRpid (default declaration, definition in SMGR_main.c)

```
SaveSMGRpid (int proc_id)
```

This function writes the process ID for the current session manager process to the lock file smgr_lock.

Argument:

proc_id: Process ID for the current session manager process (input).

Returns zero if successful or -1 if the lock file already exists and the owner of the current process does not have permission to write to the file.

SMGR_CheckJobQueue (declaration in SMGR_defs.h, definition in SMGR_job.c)

```
int SMGR_CheckJobQueue ()
```

This function is .

SMGR_DeleteJob (declaration in SMGR_defs.h, definition in SMGR_db_t.c)

```
void SMGR_DeleteJob (char *module, char *test_id)
```

This function .

Arguments:

module: (input)

test_id: (input)

SMGR_DeleteMessage (declaration in SMGR_defs.h, definition in SMGR_db_t.c)

```
void SMGR_DeleteMessage (char *module, char *test_id,  
                        char *start_time, char *end_time)
```

This function .

Arguments:

module: (input)

test_id: (input)

start_time: (input)

end_time: (input)

SMGR_EvaluateResources (declaration in SMGR_defs.h, definition in SMGR_resource.c)

```
int SMGR_EvaluateResources ()
```

This function .

SMGR_ExecuteJob (declaration in SMGR_defs.h, definition in SMGR_job.c)

```
void SMGR_ExecuteJob (SMGR_PJob job)
```

This function .

Argument:

job:

SMGR_GetJobs (declaration in SMGR_defs.h, definition in SMGR_db_t.c)

```
void SMGR_GetJobs (SMGR_Job **list_ptr, int *num_jobs)
```

This function .

Arguments:

list_ptr:

num_jobs:

SMGR_GetNextJob (declaration in SMGR_defs.h, definition in SMGR_job.c)

```
Boolean SMGR_GetNextJob (SMGR_PJob job)
```

This function .

Argument:

job:

SMGR_GetPrerequisites (declaration in SMGR_defs.h, definition in SMGR_db_t.c)

```
void SMGR_GetPrerequisites ()
```

This function .

SMGR_Initialize (declaration in SMGR_defs.h, definition in SMGR_main.c)

```
void SMGR_Initialize ()
```

This function initializes the session manager.

SMGR_InsertJob (declaration in SMGR_defs.h, definition in SMGR_db_t.c)

```
void SMGR_InsertJob (char *module, char *test_id)
```

This function .

Arguments:

module: (input)

test_id: (input)

SMGR_InsertMessage (declaration in SMGR_defs.h, definition in SMGR_db_t.c)

```
void SMGR_InsertMessage (char *module, char *test_id,  
char *start_time, char *end_time)
```

This function .

Arguments:

module: (input)

test_id: (input)

start_time: (input)

end_time: (input)

SMGR_PostJob (declaration in SMGR_defs.h, definition in SMGR_job.c)

```
void SMGR_PostJob (SMGR_PJob job)
```

This function is currently not used.

Argument:

job:

SMGR_UpdateMessageEndTime (declaration in SMGR_defs.h, definition in SMGR_db_t.c)

```
void SMGR_UpdateMessageEndTime (char *module, char *test_id,  
char *end_time)
```

This function updates the end time field of the MSG table in the SESS_MGR database when a PTDS module is completed.

Arguments:

module: Name of the module which has completed (input).

test_id: Test ID for the current test (input).

end_time: Time at which the module completed (input).

SMGR_UpdateMessageStartTime (declaration in SMGR_defs.h, definition in SMGR_db_t.c)

```
void SMGR_UpdateMessageStartTime (char *module, char *test_id,  
char *start_time)
```

This function updates the start time field of the MSG table in the SESS_MGR database when a PTDS module is started.

Arguments:

module: Name of the module which has started (input).

test_id: Test ID for the current test (input).

start_time: Time at which the module started (input).

4.7 Cflow output

```

1  main: void(), <SMGR_main.c 45>
2      IsAnotherSMGRRunning: int(), <SMGR_main.c 175>
10     SaveSMGRpid: int(), <SMGR_main.c 241>
18     init_PTDS_tekbase: void(), <tektables.c 40>
19         tbl_tekbase_init: int(), <tekbase.c 1484>
20             tbl_add_mode: void(), <tbl.c 102>
21                 tkbdone: int(), <tekbase.c 1455>
22                     tekbase_close: int(), <tekbase.c 313>
23                         clear_typecache: void(), <tekbase.c 400>
24                             tekbase_do_tql: int(), <tekbase.c 166>
25                                 query: <>
26                                     handle_error: int(), <tekbase.c 127>
27                                         query_status: <>
28                                             query_mess: <>
29                                                 query_term: <>
30                                                     query_error: <>
31                                                         tekbase_disconnect: int(), <tekbase.c 248>
32                                                             query_term: 30
33                                                                 tkbfree: int(), <tekbase.c 1431>
34                                                                     tkbupd: int(), <tekbase.c 1387>
35                                                                         check_DB: int(), <tekbase.c 345>
36                                                                             tekbase_close: 22
37                                                                                 tekbase_open: int(), <tekbase.c 276>
38                                                                                     tekbase_connect: int(), <tekbase.c 197>
39                                                                                         query_host: <>
40                                                                                             query_init: <>
41                                                                                                 handle_error: 26
42                                                                                                     query_error: 31
43                                                                                                         query_mode: <>
44                                                                                                             query_buffer: <>
45                                                                                                                 clear_typecache: 23
46                                                                                                                     clear_unique: void(), <tekbase.c 437>
47                                                                                                                         tekbase_do_tql: 24
48                                                                                                                             clear_typecache: 23
49                                                                                                             clear_unique: 49
50                                                                                             MakeUpdateList: int(), <tekbase.c 861>
51                                                                                                     MakeStringValue: char*(), <tekbase.c 727>
52                                                                                                         MakeConditionString: int(), <tekbase.c 772>
53                                                                                                             GetConditionRelop: char*(), <tekbase.c 692>
54                                                                                                                 MakeStringValue: 57
55                                                                                                                     tql_check_types: int(), <tekbase.c 490>
56                                                                                                                         query_mode: 46
57                                                                                                         get_cached_type: int(), <tekbase.c 412>
58                                                                                                             hash: unsigned int(), <tekbase.c 392>
59                                                                                                                 query: 25
60                                                                                                                     handle_error: 26
61                                                                                                                         query_error: 31

```

```

73         query_status: 27
75         cache_type: void(), <tekbases.c 428>
76         hash: 67
78         put_row: int(), <tekbases.c 558>
79         query_mode: 46
80         query_buffer: 47
85         query_mode: 46
86         tekbases_do_tql: 24
87         tkbdel: int(), <tekbases.c 1343>
88         check_DB: 36
89         MakeConditionString: 59
92         tekbases_do_tql: 24
93         tkbput: int(), <tekbases.c 1300>
94         check_DB: 36
95         tql_check_types: 64
96         MakeParameterList: int(), <tekbases.c 820>
98         query_buffer: 47
99         put_row: 78
100        query_mode: 46
102        tekbases_do_tql: 24
103        tkbget: int(), <tekbases.c 1091>
104        check_DB: 36
105        MakeConditionString: 59
106        MakeParameterList: 96
107        tql_check_types: 64
108        set_unique: int(), <tekbases.c 443>
109        tekbases_do_tql: 24
112        tekbases_do_tql: 24
113        find_col: int(), <tbl.c 170>
116        query_mode: 46
117        query_on_eob: <>
118        handle_fetch: void(), <tekbases.c 939>
121        query: 25
122        query_error: 31
123        query_status: 27
124        handle_error: 26
125        tkbcount: int(), <tekbases.c 1239>
126        check_DB: 36
127        MakeConditionString: 59
128        query_mode: 46
129        query_on_eob: 117
130        handle_count: void(), <tekbases.c 1190>
131        set_unique: 108
134        tekbases_do_tql: 24
135        query: 25
136        query_error: 31
137        query_status: 27
138        handle_error: 26
139        tbl_new: int(), <tbl.c 557>
144        DBCT_DBConnect: void(), <DBCT_utils_t.c 69>
145        tekbases_open: 39
148        SMGR_Initialize: void(), <SMGR_main.c 99>
151        RSRC_GetResourceList: void(), <RSRC_dbutils_t.c 201>
153        tbl_count: int(), <tbl.c 329>
154        find_tbl: struct*(), <tbl.c 138>
156        find_col: 113
158        parse_tbl_commands: int(), <tbl.c 209>
159        find_col: 113

```

```

163         tbl_get: int(), <tbl.c 269>
164         find_tbl: 154
165         parse_tbl_commands: 158
166         STRNG_RemoveTrailingSpaces: char*(), <STRNG_utils.c 76>
169         SMGR_GetPrerequisites: void(), <SMGR_db_t.c 205>
171         tbl_count: 153
174         tbl_get: 163
176     SMGR_EvaluateResources: int(), <SMGR_resource.c 39>
177     SMGR_CheckJobQueue: int(), <SMGR_job.c 194>
178     SMGR_GetJobs: void(), <SMGR_db_t.c 265>
180         tbl_count: 153
183         tbl_get: 163
186     RSRC_GetResources: void(), <RSRC_dbutils_t.c 137>
188         tbl_count: 153
193     RSRC_GetBits: int(), <RSRC_rlist.c 238>
194     RSRC_FindResourceName: int(), <RSRC_rlist.c 49>
196     SMGR_InsertMessage: void(), <SMGR_db_t.c 51>
198         tbl_put: int(), <tbl.c 398>
199         find_tbl: 154
200         parse_tbl_commands: 158
202     SMGR_InsertJob: void(), <SMGR_db_t.c 159>
204         tbl_put: 198
207     SMGR_GetNextJob: int(), <SMGR_job.c 44>
208     SMGR_GetJobs: 178
213     SMGR_DeleteJob: void(), <SMGR_db_t.c 183>
215         tbl_delete: int(), <tbl.c 451>
216         find_tbl: 154
217         parse_tbl_commands: 158
219     SMGR_ExecuteJob: void(), <SMGR_job.c 105>
221     RSRC_FindResourceId: int(), <RSRC_rlist.c 87>
223     RSRC_UpdateResourceBoardResourceValue: void(), <RSRC_dbutils_t.c 84>
226         tbl_get: 163
228     RSRC_GetBits: 193
229         tbl_update: int(), <tbl.c 504>
230         find_tbl: 154
231         parse_tbl_commands: 158
236     SMGR_UpdateMessageStartTime: void(), <SMGR_db_t.c 107>
238         tbl_update: 229
240     DBCT_DBSessionDisconnect: void(), <DBCT_utils_t.c 91>
241         tekbase_disconnect: 32
243     DBCT_DBConnect: 144
244     SMGR_UpdateMessageEndTime: void(), <SMGR_db_t.c 133>
246         tbl_update: 229
248     DBCT_DBDisconnect: void(), <DBCT_utils_t.c 111>
249         tekbase_disconnect: 32
251     RSRC_FreeResourceList: void(), <RSRC_rlist.c 156>
253     RemoveLockFile: void(), <SMGR_main.c 135>

```

5. Features (features)

5.1 Source Files

FEAT_main.c: The main program for features.

FEAT_dbutils_t.c:

FEAT_featureUtils.c:

FEAT_featurefits.c:

FEAT_features.c:

FEAT_fileio_t.c:

FEAT_markfile.c:

5.2 Header Files

FEAT_dbio_defs.h: Header file containing the declarations of all functions defined in FEAT_dbutils_t.c.

FEAT_featurefits.h: Header file containing the declarations of functions defined in FEAT_featurefits.c and FEAT_featureUtils.c.

FEAT_features.h: Header file containing constant definitions, macro definitions, and type definitions and the declarations of functions defined in FEAT_features.c, FEAT_featureUtils.c, and FEAT_fileio_t.c.

FEAT_jump.h: Header file containing a global variable used to save the contents of the stack at a point in the main program. The purpose is to restore all local variables in the case of a severe error such as a numerical recipes run time error. After such an error the program will use the contents of this variable to restart the program picking up with the next feature extraction command.

5.3 Defined Constants

BalancePistonCompare: value 9; defined in FEAT_features.h.

BIT_TOGGLE_MULT: value 10.0; defined in FEAT_features.h.

DBA_StartOfDataAnalysis: value -6.5; defined in FEAT_features.h.

DeltaLevelShift: value 8; defined in FEAT_features.h.

DifferentThan: value 0; defined in FEAT_features.h.

EHMS_AllPeaks: value 1; defined in FEAT_features.h.

EHMS_AverageSampleRate: Sample rate for one-second averaged data; value 1.0; defined in FEAT_features.h.

EHMS_BeginningOfTestStr: value "bot"; defined in FEAT_features.h.

EHMS_BistablePid: PID searched for spikes during periods of constant thrust having a thrust no greater than EHMS_MaxThrustForBistability; value "59"; defined in FEAT_features.h.

EHMS_CloseEnoughToZero: Range above and below zero considered to be zero by the feature extractor for purposes of determining if two data sets are the same, different, or differ by a constant offset; value 0.001; defined in FEAT_features.h.

EHMS_ControllerFullSample: Sample rate for 25 Hz data; value 0.04; defined in defined in FEAT_features.h.

EHMS_DummySigma: Used as a standard deviation for full sample data; value 1.0; defined in FEAT_features.h.

EHMS_EndOfTestStr: value "eot"; defined in FEAT_features.h.

EHMS_EngineCutoffStr: value "cutoff"; defined in FEAT_features.h.

EHMS_Erratic: value 1; defined in FEAT_features.h.

EHMS_ExpectedSigmaMultiplier: A sensor is considered erratic if the fit standard deviation is greater than this value times the expected sigma; value 4; defined in FEAT_features.h.

EHMS_FacilityFullSample: Sample rate for 50 Hz data; value 0.02; defined in FEAT_features.h.

EHMS_FullSample: value 0; defined in FEAT_features.h.

EHMS_GoodFitFactor: This value times the number of degrees of freedom is the limit of what is considered an acceptable fit according to the rule of thumb from statistics theory; value 4; defined in FEAT_features.h.

EHMS_IndeterminateThrustLevelStr: Used as the thrust_level tag for features which may span more than one thrust level; value "-999.0"; defined in FEAT_features.h.

EHMS_LOXSealEquilibrium: Thermal equilibrium reached; value 200.0; defined in FEAT_features.h.

EHMS_LOXSealEquilibriumStr: value "lox_eq"; defined in FEAT_features.h.

EHMS_LPOTPDIschargePressureA: value "209"; defined in FEAT_features.h.

EHMS_MaxCheckTypeStr: Size in bytes of maximum redline check type string; value 11; defined in FEAT_features.h.

EHMS_MaxLimitTypeStr: Size in bytes of maximum limit type string; value 6; defined in FEAT_features.h.

EHMS_MaxSettlingTime: Time to account for transients resulting from response to a change in commanded throttle; value 5.0; defined in FEAT_features.h.

EHMS_MaxSlopeForLevelShift: Maximum allowable slope for periods of constant data that bound a level shift; value 0.15; defined in FEAT_features.h.

EHMS_MaxThrustForBistability: Maximum value for thrust level for periods of constant thrust to be examined for bistability; value 65; defined in FEAT_features.h.

EHMS_MinConstantThrustPeriod: Width in seconds of smallest interval of constant thrust which will be considered by the program; value 6.0; defined in FEAT_features.h.

EHMS_MinPeriodOfLinearTankPressure: Minimum duration of a period of linear LPOTP discharge pressure during which to look for erratic behaviour in a sensor trace (in seconds); value 6; defined in FEAT_features.h.

EHMS_MinSettlingTime: Time to account for transients resulting from response to a change in commanded throttle; value 1.0; defined in FEAT_features.h.

EHMS_MinThrustForBistability: Minimum value for thrust level for periods of constant thrust to be examined for bistability; value 0; defined in FEAT_features.h.

EHMS_MinThrustSlope: Empirically determined constant; value 0.4; defined in FEAT_features.h.

EHMS_NonErratic: value 0; defined in FEAT_features.h.

EHMS_NumFeatureExtractionModules: value 14; defined in FEAT_features.h.

EHMS_NumSigmaForSigLevelShift: The delta between two periods of constant data must exceed this value times sigma_mag of the first data set to be deemed significant; value 3; defined in FEAT_features.h.

EHMS_NumSigmasForFlat: Number of sigmas above/below 0.0 the slope of a fitted line must lie to have the line considered flat (slope of 0.0); value 3.0; defined in FEAT_features.h.

EHMS_NumSigmasForSpike: Number of sigmas beyond fit which a point must exhibit to qualify it as a peak, having a magnitude outside the noise level; value 4.0; defined in FEAT_features.h.

EHMS_OneSecondAve: value 1; defined in FEAT_features.h.

EHMS_PctAreaToCheck: Percentage of thrust period to check for bistability (furthest from end points); value 0.85; defined in FEAT_features.h.

EHMS_PrimaryPeak: value 0; defined in FEAT_features.h.

EHMS_SameAsProbability: Probability cutoff above which two compared by the kstwo function are considered drawn from the same; value 0.70; defined in FEAT_features.h.

EHMS_ScaledMinThrustSlope: value 0.00000001; defined in FEAT_features.h.

EHMS_SigmaFudgeNumber: Fake number used as a standard deviation when the number is really 0.0 because the *Numerical Recipes* code cannot handle 0.0; value 0.02; defined in FEAT_features.h.

EHMS_SigmasForSigStep: This value times sigma[i] must be exceeded for the slope of a level shift to be significant; value 2; defined in FEAT_features.h.

EHMS_SmallestPossibleStepSize: Smallest step size by which any digital data measured; value 0.0001; defined in FEAT_features.h.

EHMS_SpikeCountForBistability: Number of spikes needed on an interval before the PBP is judged to be bistable at that thrust level; value 2; defined in FEAT_features.h.

EHMS_SpikeWidth: Width in one second intervals of a spike; value 1; defined in FEAT_features.h.

EHMS_SubIntervalLength: Length in seconds of subinterval used in finding level shifts; value 6; defined in FEAT_features.h.

EHMS_ThrustPid: PID to be used for determining periods of constant thrust; value "287"; defined in FEAT_features.h.

EHMS_TurbineSealEquilibrium: Thermal equilibrium reached; value 150.0; defined in FEAT_features.h.

EHMS_TurbineSealEquilibriumStr: value "ts_eq"; defined in FEAT_features.h.

executed: value 0; defined in FEAT_features.h.

FEAT_MainDB: value "ssme_data"; defined in FEAT_features.h.

FEAT_MaxPathLength: value 60; defined in FEAT_features.h.

FEAT_OPOV_bit_toggle: Used in bistability calculations; value 0.07; defined in FEAT_features.h.

FEAT_SQLScriptDir: value "FEAT_SQL_SCRIPTS"; defined in FEAT_features.h.

FEAT_TestEnvVar: value "TEST_DB"; defined in FEAT_features.h.

FindBistable: value 3; defined in FEAT_features.h.
 FindDrift: value 10; defined in FEAT_features.h.
 FindErraticBehaviour: value 1; defined in FEAT_features.h.
 FindLevelShift: value 5; defined in FEAT_features.h.
 FindPeak: value 2; defined in FEAT_features.h.
 FindSpike: value 4; defined in FEAT_features.h.
 IsFlat: value 7; defined in FEAT_features.h.
 NoisyPid: value 11; defined in FEAT_features.h.
 not_executed: value 1; defined in FEAT_features.h.
 NUMSIGMAS: value 3.0; defined in FEAT_features.h.
 PERCENTAGE_RANGE: value 0.070; defined in FEAT_features.h.
 PWR_LEVEL_OFFSET: Time added to the beginning of a power level to try to avoid noise; value 1.4; defined in FEAT_features.h.
 RedlineCheck: value 6; defined in FEAT_features.h.
 RedundChannelCheck: value 12; defined in FEAT_features.h.
 YNORM: value 100.0; defined in FEAT_features.h.
 ZeroShiftCheck: value 13; defined in FEAT_features.h.

5.4 Defined Types

EHMS_ConstantPeriodID, EHMS_PConstantPeriodID (defined in FEAT_features.h)

```

typedef struct EHMS_constantperiodid {
    float
        start_time,
        end_time,
        magnitude,
        mag_sigma,
        slope;
} EHMS_ConstantPeriodID, *EHMS_PConstantPeriodID;

```

Members:

start_time: Start time of period of constant thrust.
 end_time: End time of period of constant thrust.
 magnitude:
 mag_sigma:
 slope:

EHMS_DataIDRecord, EHMS_PDataIDRecord (defined in FEAT_features.h)

```

typedef struct EHMS_dataidrecord {
    char
        expert[DBFL_MaxExpertModuleNameLength + 1],

```

```

    test_str[DBFL_MaxTestIdLength + 1],
    pid_str[DBFL_PidNameLength + 1],
    descrip[DBFL_MaxMeasurementStringLength + 1],
    compare_test[DBFL_MaxTestIdLength + 1],
    compare_pid[DBFL_PidNameLength + 1],
    compare_descrip[DBFL_MaxMeasurementStringLength + 1];
    float
        start_time,
        end_time,
        unaltered_start_time,
        unaltered_end_time,
        period,
        thrust_level;
} EHMS_DataIDRecord, *EHMS_PDataIDRecord;

```

Members:

expert:
 test_str: Test ID of the source of the data.
 pid_str: PID from the specified test.
 descrip: Description of the measurement.
 compare_test: Test ID of the comparison test used.
 compare_pid: PID from the specified comparison test.
 compare_descrip: Description of the measurement from the comparison test.
 start_time: Start time of the feature;
 end_time: End time of the features.
 unaltered_start_time:
 unaltered_end_time:
 period: Sample period for the data set.
 thrust_level: Thrust level over the given time range.

EHMS_DifferentThanRecord, EHMS_PDifferentThanRecord (defined in
FEAT_features.h)

```

typedef struct EHMx_differentthanrecord {
    float
        chi_square,
        prob;
    FEAT_Boolean
        coeffs_within_errorBars,
        differ_by_constant_offset;
    float
        offset,
        offset_sigma;
} EHMS_DifferentThanRecord, *EHMS_PDifferentThanRecord;

```

Members:

chi_square:
 prob:

coeffs_within_error_bars:
 differ_by_constant_offset:
 offset:
 offset_sigma:

EHMS_FeatureRecord, EHMS_PFeatureRecord (defined in FEAT_features.h)

```

typedef struct EHMS_featurerecord {
    struct EHMS_featurerecord
        *next,
        *prev;
    char
        test_id[DBFL_MaxTestIdLength + 1],
        sensor[DBFL_PidNameLength + 1];
    float
        peak_ht,
        taph,
        fwhm,
        tafwhm1,
        tafwhm2;
    EHMS_FitType
        fit_type;
    float
        chi_square;
    int
        num_params;
    float
        offset,
        *fitted_params;
} EHMS_FeatureRecord, *EHMS_PFeatureRecord;
  
```

Members:

next: Pointer to the next feature record.
 prev: Pointer to the previous feature record.
 test_id:
 sensor: Sensor from which the data was taken.
 peak_ht: Maximum peak height.
 taph: Time at the maximum peak height.
 fwhm: Full width half maximum for peak (feature).
 tafwhm1: Time at full width half maximum 1.
 tafwhm2: Time at full width half maximum 2.
 fit_type: Type of equation fit to points comprising feature.
 chi_square: Goodness of fit measurement.
 num_params: Number of elements in the a_{ks} array.
 offset: Number of elements in the a_{ks} array.
 fitted_params: Coefficients of the fit from which function parameters may be calculated.

EHMS_FeatureRecordHead, EHMS_PFeatureRecordHead (defined in FEAT_features.h)

```
typedef struct EHMS_featurerecordhead {
    EHMS_PFeatureRecord
    first,
    last;
} EHMS_FeatureRecordHead, *EHMS_PFeatureRecordHead;
```

Members:

first: First feature in the list.

last: Last feature in the list.

EHMS_FitType (defined in FEAT_features.h)

```
typedef enum EHMS_FitFuncs
    {Gaussian, FastRiseExpFall, NthOrderPoly} EHMS_FitType;
```

EHMS_LinearLPOTPDDischargePressureRec,

EHMS_PLinearLPOTPDDischargePressureRec (defined in FEAT_features.h)

```
typedef struct EHMS_linearLPOTPDdischargepressurerec {
    float
    start_time,
    end_time;
} EHMS_LinearLPOTPDDischargePressureRec,
*EHMS_PLinearLPOTPDDischargePressureRec;
```

Members:

start_time:

end_time:

EHMS_RedlineCheckType (defined in FEAT_features.h)

```
typedef enum EHMS_RedCheckChoice
    {single_pid, both_pids, either_pid, difference}
    EHMS_RedlineCheckType;
```

EHMS_RedlineInfoRecord, EHMS_PRedlineInfoRecord (defined in FEAT_features.h)

```
typedef struct EHMS_redlineinfo record {
    float
    start_time,
    end_time,
    redline,
    redline_time;
} EHMS_RedlineInfoRecord, *EHMS_PRedlineInfoRecord;
```

Members:

start_time:

end_time:

redline:

redline_time:

EHMS_RedlineType (defined in FEAT_features.h)

```
typedef enum EHMS_RedlineTypes {upper, lower} EHMS_RedlineType;
```

EHMS_SpikeType (defined in FEAT_features.h)

```
typedef enum EHMS_Spikes {positive, negative} EHMS_FitType;
```

EHMS_ThrustPeriodID, EHMS_PThrustPeriodID (defined in FEAT_features.h)

```
typedef struct EHMS_thrustperiodid {  
    float  
        start_time,  
        end_time,  
        thrust_level;  
} EHMS_ThrustPeriodID, *EHMS_PThrustPeriodID;
```

Members:

start_time: Start time of period of constant thrust.

end_time: End time of period of constant thrust.

thrust_level:

FEAT_Boolean (defined in FEAT_features.h)

```
typedef enum feat_boolean {Ffalse, Ftrue} FEAT_Boolean;
```

5.5 Global Variables

EHMS_SelectString: char [DBFL_MaxCommandLength]; defined in FEAT_features.h.

5.6 Functions

calc_stddev (definition in FEAT_featureUtils.c)

```
double calc_stddev (float *data, int num_pts)
```

covsrt (declaration in FEAT_featurefits.h, definition in FEAT_featurefits.c)

```
void covsrt (float **covar, int ma, int lista[], int mfit)
```

DBA_StandardStringToPid (declaration in FEAT_dbio_defs.h, definition in FEAT_dbutils_t.c)

```
void DBA_StandardStringToPid (char *test_id, char *pid_name,  
                             char *standard_string)
```

DBA_TimeRangeToRecNumRange (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
void DBA_TimeRangeToRecNumRange (float start_time, float end_time,  
                                float sample_period,  
                                int *start_rec, int *end_rec)
```

DBIO_FetchRedlineInfo (declaration in FEAT_dbio_defs.h, definition in FEAT_dbutils_t.c)

```
void DBIO_FetchRedlineInfo (char *test_id, char *pid,
                             EHMS_RedlineType limit_type,
                             EHMS_PRedlineInfoRecord *red_info_rec,
                             int *num_recs)
```

DBIO_GetConstantThrustLevels (declaration in FEAT_dbio_defs.h, definition in FEAT_dbutils_t.c)

```
void
  DBIO_GetConstantThrustLevels (char *test, int *num_thrust_ids,
                                EHMS_PThrustPeriodID *thrust_ids)
```

DBIO_GetPidInfo (declaration in FEAT_dbio_defs.h, definition in FEAT_dbutils_t.c)

```
void DBIO_GetPidInfo (char *test, char *pid, char *units,
                      char *descrip, float *start, float *end,
                      float *rate)
```

DBIO_GetTheFeatureExtractionCommands (declaration in FEAT_dbio_defs.h, definition in FEAT_dbutils_t.c)

```
char **DBIO_GetTheFeatureExtractionCommands (int *num_entries)
```

DBIO_PidToSensorLabel (declaration in FEAT_dbio_defs.h, definition in FEAT_dbutils_t.c)

```
char *DBIO_PidToSensorLabel (char *test, char *pid)
```

EHMS_AddConstantPeriodID (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
void EHMS_AddConstantPeriodID (EHMS_PConstantPeriodID *period_ids,
                                int num_period_ids,
                                float start_time, float end_time,
                                float magnitude, float mag_sigma,
                                float slope)
```

EHMS_AddFeatureRecord (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
void EHMS_AddFeatureRecord (EHMS_PFeatureRecord new_data_record,
                             EHMS_PFeatureRecordHead
                             data_record_head)
```

EHMS_AddLinearPressureRec (definition in FEAT_featureUtils.c)

```
void EHMS_AddLinearPressureRec
  (EHMS_PLinearLPOTPDDischargePressureRec *pressure_recs,
   int *num_linear_periods, float start_time, float end_time)
```

EHMS_AddThrustPeriodID (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
void EHMS_AddThrustPeriodID (EHMS_PThrustPeriodID *thrust_ids,
                             int num_thrust_ids, float start_time,
                             float end_time, float thrust_level)
```

EHMS_AnalyzeThrustProfile (declaration in FEAT_features.h, definition in FEAT_features.c)

```
EHMS_PThrustPeriodID
EHMS_AnalyzeThrustProfile (char *test_id, int *num_thrust_ids,
                           char *db_string)
```

EHMS_ArrayAdd (definition in FEAT_featureUtils.c)

```
int EHMS_ArrayAdd (float *data, int num_pts, float val)
```

EHMS_ArrayMult (definition in FEAT_featureUtils.c)

```
int EHMS_ArrayMult (float *data, int num_pts, float mult)
```

EHMS_BalancePistonCheckInit (declaration in FEAT_features.h, definition in FEAT_features.c)

```
FEAT_Boolean
EHMS_BalancePistonCheckInit (EHMS_PDataIDRecord data_id,
                             float *time1, float *data1,
                             float *sigmas1, float **time2,
                             float **composite_data1,
                             float **composite_sigmas1,
                             float **composite_data2,
                             float **composite_sigmas2)
```

This function is the primary function of the BalancePistonCompare module, which serves as a front end preprocessor to another standard module, DifferentThan. BalancePistonCompare produces a composite data set made up of the point for point difference between the two specified data sets. BalancePistonCompare creates two composite data sets, one with data drawn from the current test and one with data drawn from the comparison test. A call is then made to the DifferentThan module with the composite data sets. The purpose of this specialized module is to look for changes in the net force exerted on the balance piston between tests at similar thrust levels.

Arguments:

```
data_id:
time1:
data1:
sigmas1:
time2:
composite_data1:
composite_sigmas1:
composite_data2:
composite_sigmas2:
```


Command table inputs:

expert:
sensor:
sensor_postfix:
modulename:
starttime:
endtime:
compare_descrip (param1):
compare_test (param2):
num_comparison_sigmas (param3):

Any features found by this module are reported to the same table used by the standard DifferentThan module, F_DIFTHA. The results of this module are distinguishable by entries in the sensor and comparison sensor columns which, for BalancePistonCompare, are of the form PID#-PID#, which correspond to the sensor and comparison sensor.

EHMS_BuildRunTimeCommandArray (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
char
    **EHMS_BuildRunTimeCommandArray (char *test_id,
                                      char **command_table_rows,
                                      int num_entries,
                                      EHMS_PThrustPeriodID thrust_ids,
                                      int num_thrust_ids,
                                      int *num_rt_commands)
```

EHMS_BuildSelectString (definition in FEAT_featureUtils.c)

```
void EHMS_BuildSelectString (char *select_string, char *test_id,
                             char *pid, int start_rec,
                             int end_rec)
```

EHMS_CompareRealNums (definition in FEAT_featureUtils.c)

```
FEAT_Boolean EHMS_CompareRealNums (float arg1, float arg2,
                                    float precision)
```

EHMS_CopyDataID (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
void EHMS_CopyDataID (EHMS_PDataIDRecord new_data_id,
                      EHMS_PDataIDRecord old_data_id)
```

EHMS_CreateConstantPeriodRecord (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
EHMS_PConstantPeriodID EHMS_CreateConstantPeriodRecord ()
```

EHMS_CreateDataIDRecord (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
EHMS_PDataIDRecord EHMS_CreateDataIDRecord ()
```

EHMS_CreateDifferentThanRecord (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

EHMS_PDifferentThanRecord EHMS_CreateDifferentThanRecord ()

EHMS_CreateFeatureRecord (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

EHMS_PFeatureRecord
EHMS_CreateFeatureRecord (EHMS_PDataIDRecord data_id)

EHMS_CreateFeatureRecordHead (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

EHMS_PFeatureRecordHead EHMS_CreateFeatureRecordHead ()

EHMS_DeleteFeatureRecord (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

void
EHMS_DeleteFeatureRecord (EHMS_PFeatureRecord old_data_record,
EHMS_PFeatureRecordHead
data_record_head)

EHMS_DeltaDifferentThan (declaration in FEAT_features.h, definition in FEAT_features.c)

void
EHMS_DeltaDifferentThan (EHMS_PDataIDRecord data_id,
float *time1, float *data1,
float *sigmas1, float *time2,
float *data2, float *sigmas2,
int num_comparison_sigmas,
EHMS_PDifferentThanRecord test_results,
int num_points, float step_size)

This function is the primary function of the DifferentThan module, which analyzes data from two sensors which may be drawn from the current test or the current test and a comparison test determined at run time.

Arguments:

data_id:
time1:
data1:
sigmas1:
time2:
data2:
sigmas2:
num_comparison_sigmas:
test_results:
num_points:

step_size1:

Command table inputs:

expert:

sensor:

sensor_postfix:

modulename:

starttime:

endtime:

compare_descrip (param1):

polyorder (param2):

num_comparison_sigmas (param3):

Any features found are reported to the table F_DIFTHA.

EHMS_DeltaLevelShift (declaration in FEAT_features.h, definition in FEAT_features.c)

FEAT_Boolean

```
EHMS_DeltaLevelShift (EHMS_PDataIDRecord data_id, float *time1,
                      float *data1, float *sigmas1,
                      float *time2, float *data2,
                      float *sigmas2, int num_points,
                      EHMS_PLinearLPOTPDDischargePressureRec
                      pressure_rec, int num_linear_periods,
                      float threshold, int SubIntervalLength)
```

This function is the primary function of the DeltaLevelShift module, which serves as a front end to the LevelShift module. The purpose of this module is to produce a composite data set made up of the point for point difference between the two specified sensor data sets. A call is then made to the LevelShift module with the composite data set. Using this preprocessing allows the expert to look for level shifts in the difference between two pids. This is often useful in such cases as balance piston analysis where changes in the net force exerted on the balance piston can be detected by looking for level shifts in the data set comprised of pid 327 - pid 328.

Arguments:

data_id:

time1:

data1:

sigmas1:

time2:

data2:

sigmas2:

num_points:

pressure_rec:

num_linear_periods:

threshold:

SubIntervalLength:

Command table inputs:

expert:
sensor:
sensor_postfix:
modulename:
starttime:
endtime:
compare_descrip (param1):

Any features found by this module are reported to the same table used by the standard LevelShift module, F_LEVSH. The results of this module are distinguishable by an entry in the sensor column of the format PID#-PID#.

EHMS_DescribeConstantPeriod (definition in FEAT_features.c)

```
void
EHMS_DescribeConstantPeriod (EHMS_PConstantPeriodID period_id,
                             float *xarray, float *data,
                             float *sigmas, int num_pts)
```

EHMS_DescribeConstantThrustLevel (declaration in FEAT_features.h, definition in FEAT_features.c)

```
int EHMS_DescribeConstantThrustLevel (EHMS_PDataIDRecord data_id,
                                       float *time, float *data,
                                       float *sigmas)
```

EHMS_DetectBistability (declaration in FEAT_features.h, definition in FEAT_features.c)

```
FEAT_Boolean EHMS_DetectBistability (EHMS_PDataIDRecord data_id,
                                       float *time, float *data,
                                       float *sigmas,
                                       int num_data_pts)
```

This is the primary function of the special purpose module, FindBistable, intended to test for the presence of Preburner Pump Bistability in the SSME. This goal is accomplished by searching the pid defined by the variable EHMS_BistablePid for negative going spikes over each period of constant thrust having a thrust level of EHMS_MinThrustForBistability or lower. The method used to detect spikes on the interval of interest is the same as that applied by the FindSpike module, with the exclusion of second order fitting capability. If the number of spikes found on any given interval is greater than the number defined by the constant EHMS_SpikeCountForBistability, then that interval is flagged as containing an instance of Preburner Pump Bistability.

Arguments:

data_id:
time:
data:
sigmas:
num_data_points:

Command table inputs:

expert:
sensor:
sensor_postfix:
modulename:
starttime:
endtime:

Any features found are reported to the table F_BISTAB.

EHMS_ExecuteTheLoadScripts (definition in FEAT_fileio_t.c)

```
void EHMS_ExecuteTheLoadScripts (FEAT_Boolean *modules_executed,  
                                char *db_string)
```

EHMS_FastRiseExpFall (declaration in FEAT_featurefits.h, definition in FEAT_featurefits.c)

```
void EHMS_FastRiseExpFall (float x, float *a, float *y,  
                           float *dyda, int na)
```

EHMS_FeatureExtractor (declaration in FEAT_features.h, definition in FEAT_features.c)

```
FEAT_Boolean EHMS_FeatureExtractor (char *test_id,  
                                    char *table_entry_str,  
                                    int *module)
```

EHMS_FindConstantThrust (declaration in FEAT_features.h, definition in FEAT_features.c)

```
EHMS_PThrustPeriodID  
EHMS_FindConstantThrust (EHMS_PDataIDRecord data_id,  
                        float *time, float *data,  
                        float *sigmas, int num_data_pts,  
                        int *num_thrust_ids)
```

EHMS_FindDrift (definition in FEAT_features.c)

```
FEAT_Boolean  
EHMS_FindDrift (EHMS_PDataIDRecord data_id, float *time,  
               float *data, float *smoothed_data,  
               float *sigmas, int n, float multiplier,  
               float time_range,  
               EHMS_PLinearLPOTPDDischargePressureRec  
               pressure_rec, int num_linear_periods)
```

EHMS_FindErraticBehaviour (declaration in FEAT_features.h, definition in FEAT_features.c)

```
FEAT_Boolean  
EHMS_FindErraticBehaviour (EHMS_PDataIDRecord data_id,  
                           float *time, float *data,  
                           float *sigmas, float expected_sigma,
```



```

int num_points,
EHMS_PLinearLPOTPDDischargePressureRec
pressure_rec, int num_linear_periods,
float step_size)

```

This is the primary function of the FindErraticBehaviour module. Processing begins by comparing the number of points on the interval of interest to the number of parameters varied in a second order fit multiplied by the value of EHMS_GoodFitFactor. If the number of points is less than the result of the calculation described above, a first order fit is used instead of a second order. It was found that for short time intervals this technique produced better determinations of erratic behavior. After the fit has been made the standard deviation calculated for the fit is adjusted to account for bit toggle and then compared against an expected value, which is specified in the command table as PARAM1. If the standard deviation for the fit exceeds this value, then the sensor trace is deemed to be erratic.

Arguments:

```

data_id:
time:
data:
sigmas:
expected_sigma:
num_points:
pressure_rec:
num_linear_periods:
step_size:

```

Command table inputs:

```

expert:
sensor:
sensor_postfix:
modulename:
starttime:
endtime:
expected_sigma (param1):

```

Any features found are reported to the table F_ERRAT.

EHMS_FindInflectionPtsByLinearTrendRemoval (definition in FEAT_featureUtils.c)

```

void EHMS_FindInflectionPtsByLinearTrendRemoval
(float *time, float *data, float *sigmas, int num_pts,
EHMS_PLinearLPOTPDDischargePressureRec *pressure_recs,
int *num_linear_periods, float threshold)

```

This routine operates by first checking the data period (initially time zero to engine shutdown) to be sure the time span is greater than EHMS_MinPeriodOfLinearTankPressure seconds. The time slice must meet or exceed this requirement in order for the period to be valid for feature detection. Secondly any linear trend in the data is removed. This results in the start and end points being mapped to a magnitude of zero. The resulting data set is then searched for the point of maximum magnitude. If this maximum is greater than four times the standard deviation at that point, then the period is broken into two subintervals at the point of maximum magnitude and the routine recursively called for each

subinterval. If the maximum magnitude is not greater than four times the standard deviation, then the interval is considered devoid of prominent peaks and is reported as a single continuous interval for the purposes of feature extraction. When all levels of the routine have returned due to reduction of subintervals to less than EHMS_MinPeriodOfLinearTankPressure seconds or lack of prominent peaks, the number of collected subintervals is returned as well as an array of start and end times for the periods of linear behavior.

EHMS_FindLevelShift (declaration in FEAT_features.h, definition in FEAT_features.c)

```
FEAT_Boolean
EHMS_FindLevelShift (EHMS_PDataIDRecord data_id, float *time,
                    float *data, float *sigmas,
                    float data_range, int num_points,
                    EHMS_PLinearLPOTPDDischargePressureRec
                    pressure_rec, int num_linear_periods,
                    int SubIntervalLength )
```

This is the primary function of the module FindLevelShift, which has as its purpose to detect changes in a sensor trace from one constant value to another. It works by breaking the specified time period into sub periods of EHMS_SubIntervalLength seconds and making a separate first order polynomial fit to each period. The average and standard deviation of the constant offset terms of the fits are used to locate any level shifts. Any significant change in the data will manifest itself by a slope change in the set of fits to the data. A dramatic slope change will result in a line having a projection on the Y axis which far exceeds the average as determined from the multiple fits. Any excursions which exceed the average value plus or minus 3 times the standard deviation indicate the start of a level shift. Excursions are tracked until such time as they return to within acceptable limits or the period of interest is exhausted.

Arguments:

```
data_id:
time:
data:
sigmas:
data_range:
num_points:
pressure_rec:
num_linear_periods:
SubIntervalLength:
```

Command table inputs:

```
expert:
sensor:
sensor_postfix:
modulename:
starttime:
endtime:
```

Any features found are reported to the table F_LEVSH.

EHMS_FindLinearLPOTPDIschargePressure (definition in FEAT_featureUtils.c)

```
EHMS_PLinearLPOTPDIschargePressureRec
    EHMS_FindLinearLPOTPDIschargePressure (char *pid,
                                           float start_time,
                                           float end_time,
                                           char *test_id,
                                           int *num_linear_periods)
```

EHMS_FindPeak (declaration in FEAT_features.h, definition in FEAT_features.c)

```
FEAT_Boolean EHMS_FindPeak (EHMS_PDataIDRecord data_id,
                             float *time, float *data,
                             float *sigmas, int num_data_pts,
                             int peak_set, float min_peak,
                             float min_width)
```

This function is the primary function of the FindPeak module. This module employs a technique where transitions in the data are identified by moving a tangent line over the data and noting the slope of the line. A line made up of three points is used, the center point being the "tangent" point. If the slope of this line exceeds a limit which allows for noise, then the time at the tangent point is marked as a transition time. Identification of a transition point is made by a check for slope > or < 3.0 times the standard deviation at the tangent point. The reasoning behind this is that the worst case slope which is still classified as noise will be

$$\text{slope} = (4 * \text{sigma}) / 2.0 = 2 * \text{sigma}$$

The slope of the tangent line must exceed $2 * \text{sigma}$ in order to be considered above the noise level in the data. Good results have been obtained using $3 * \text{sigma}$.

The sign of the slope is also exploited for peak detection. Changes in sign indicate a cusp or peak in the sensor data. Only positive going peaks are detected, as indicated by a change in the sign of the slope from positive to negative. A peak is tracked from the first excursion outside the noise level to until either the measurement levels off, or begins to exhibit another peak. The detected peaks are then checked to make sure that they have a magnitude $\geq$ the minimum specified in the command table and that they have a width $\geq$ that specified in the command table. The data is then fitted to one of two models based on the position of the cusp relative to the start and end time of the peak feature. The first model is a fast rise with an exponential falloff. The second model is a gaussian curve. Those peaks with a chi-square per degree of freedom value of greater than EHMS_ChiSquareFactor are discarded. The remaining peaks are then described and categorized as features. If it was specified in the command table that only the primary peak be reported, then all others are discarded.

Arguments:

data_id:
time:
data:
sigmas:
num_data_points:
peak_set:
min_peak:
min_width:

Command table inputs:

expert:
 sensor:
 sensor_postfix:
 modulename:
 starttime:
 endtime:
 peak_set (param1):
 min_peak (param2):
 min_width (param3):

Any features found are reported to the table F_PEAK.

EHMS_FindSpike (declaration in FEAT_features.h, definition in FEAT_features.c)

```

FEAT_Boolean
  EHMS_FindSpike (EHMS_PDataIDRecord data_id, float *time,
                  float *data, float *sigmas, int num_points,
                  EHMS_PLinearLPOTPDDischargePressureRec
                  pressure_rec, int num_linear_periods,
                  float step_size)
  
```

The module FindSpike is a variation on FindErraticBehaviour. It makes a fit to the data using a second order polynomial form, but for each data point on the interval a check is made for points which fall outside the limit defined by

$$\text{fitted_point} \pm (\text{EHMS_NumSigmasForSpike} * \text{fit_std_dev})$$

where fit_std_dev is the standard deviation calculated for the fit, adjusted for bit toggles as shown below:

$$\text{ftd_std_dev} = \text{fit_std_dev} + (0.5 * \text{step_size})$$

Any excursions outside this limit, which exists for no longer than EHMS_SpikeWidth seconds, are identified as spikes. The magnitude of the spike is reported as magnitude. The sign of magnitude is determined by the convention

$$\text{raw_data}[\text{Index_of_peak}] - \text{fitted_point}[\text{Index_of_peak}]$$

Arguments:

data_id:
 time:
 data:
 sigmas:
 num_points:
 pressure_rec:
 num_linear_periods:
 step_size:

Command table inputs:

expert:
 sensor:

sensor_postfix:
 modulename:
 starttime:
 endtime:

Any features found are reported to the table F_SPIKE.

EHMS_FindSpike_a (definition in FEAT_features.c)

```
FEAT_Boolean EHMS_FindSpike_a (EHMS_PDataIDRecord data_id,
                                float *time, float *data,
                                float *smoothed_data,
                                float *sigmas, int num_points,
                                float bit_toggle, float multiplier)
```

EHMS_FindSpike_b (definition in FEAT_features.c)

```
FEAT_Boolean EHMS_FindSpike_b (EHMS_PDataIDRecord data_id,
                                struct SSME_data *data,
                                float *time, float bit_toggle,
                                float pid_range, float rate)
```

EHMS_FindStepSize (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
float EHMS_FindStepSize (char *test_id, char *pid,
                          float sample_period)
```

EHMS_FreeMRQMemory (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
void EHMS_FreeMRQMemory (int *lista, float **covar, float **alpha,
                          int ma)
```

EHMS_FreePeakRecords (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
void EHMS_FreePeakRecords (EHMS_PFeatureRecordHead list_head)
```

EHMS_FullSamplePid (definition in FEAT_featureUtils.c)

```
FEAT_Boolean EHMS_FullSamplePid (char *pid_str)
```

EHMS_Gaussian (declaration in FEAT_featurefits.h, definition in FEAT_featurefits.c)

```
void EHMS_Gaussian (float x, float *a, float *y, float *dyda,
                    int na)
```

EHMS_GetCompositeDataSet (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
float **EHMS_GetCompositeDataSet (float *data1, float *sigmas1,
                                   float *data2, float *sigmas2,
                                   int num_points,
                                   EHMS_RedlineCheckType
```

```

                                check_type)

EHMS_GetEngineCutoff    (declaration in FEAT_dbio_defs.h, definition in
FEAT_dbutils_t.c)

    float EHMS_GetEngineCutoff (char *test)

EHMS_GetFitInterval    (declaration in FEAT_features.h, definition in
FEAT_featureUtils.c)

    void EHMS_GetFitInterval (float *data, int num_data_pts, int *i,
                                int *j, float period)

EHMS_GetFitIntervalForSpikeCheck (declaration in FEAT_features.h, definition in
FEAT_featureUtils.c)

    void EHMS_GetFitIntervalForSpikeCheck (float *data, int *i,
                                            int *j, float period)

EHMS_GetMRQMemory (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

    void EHMS_GetMRQMemory (float **a_ptr, int **lista_ptr,
                            float ***covar_ptr, float ***alpha_ptr,
                            int ma)

EHMS_GetNumberBasisFuncs (declaration in FEAT_featurefits.h, definition in
FEAT_featurefits.c)

    int EHMS_GetNumberBasisFuncs (void (*fit_func)(), int poly_order)

EHMS_GetSigmas (declaration in FEAT_features.h)

    (no longer defined????)

EHMS_GetSmoothedOffset (definition in FEAT_featureUtils.c)

    int EHMS_GetSmoothedOffset (float *smoothed_time, float time,
                                int num_pts)

EHMS_GetThrustLevel    (declaration in FEAT_features.h, definition in
FEAT_featureUtils.c)

    float EHMS_GetThrustLevel (EHMS_PThrustPeriodID thrust_ids,
                                int num_thrust_ids, float start_time,
                                float end_time)

EHMS_GetThrustLevelIntersection (declaration in FEAT_features.h, definition in
FEAT_featureUtils.c)

    FEAT_Boolean
    EHMS_GetThrustLevelIntersection (EHMS_PDataIDRecord data_id)

EHMS_IsFlat (declaration in FEAT_features.h, definition in FEAT_features.c)

    FEAT_Boolean

```



```
EHMS_IsFlat (EHMS_PDataIDRecord data_id, float *time,
             float *data, float *sigmas,
             EHMS_PLinearLPOTPDDischargePressureRec pressure_rec,
             int num_linear_periods, int num_points)
```

EHMS_MakeDummySigmas (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
float *EHMS_MakeDummySigmas (int num_pts)
```

EHMS_MakeFit (declaration in FEAT_featurefits.h, definition in FEAT_featurefits.c)

```
void EHMS_MakeFit (float *time, float *data, float *sigmas,
                  int num_pts, float *a, int ma, int *lista,
                  int mfit, float **covar, float **alpha,
                  float *chi_square, void (*fit_func)(),
                  float *alamda, float *fit_std_dev)
```

This curvefitting routine is the basis of all the feature extraction modules. It makes use of the routine mrqmin. EHMS_MakeFit can fit to any function that is differentiable with respect to the fitted parameters over the interval of interest. Each model for use in curve fitting exists as an independent subroutine. A pointer to the desired model for fitting is passed to the routine, indicating which model (functional form) to fit the data to. Currently models exist for an Nth order polynomial, a fast rising function with an exponential fall-off, as well as for a Gaussian (bell) curve. Additional models can be added to the system by writing a short model routine based on the existing examples and making an entry in the routine EHMS_GetNumBasisFuncs to note how many fitted parameters are involved.

EHMS_MatchPostfix (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
void EHMS_MatchPostfix (char *postfix, char *pid)
```

EHMS_MaxMin (definition in FEAT_featureUtils.c)

```
int EHMS_MaxMin (float *maxval, float *minval, float *data,
                 int num_pts)
```

EHMS_NoisyPid (definition in FEAT_features.c)

```
FEAT_Boolean EHMS_NoisyPid (EHMS_PDataIDRecord data_id,
                             float *time, float *sigmas,
                             int num_points, float gross_sigma,
                             float fine_sigma)
```

EHMS_NthOrderPoly (declaration in FEAT_featurefits.h, definition in FEAT_featurefits.c)

```
void EHMS_NthOrderPoly (float x, float *a, float *y, float *dyda,
                        int na)
```

EHMS_ParseTableEntry (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
FEAT_Boolean EHMS_ParseTableEntry (char *table_entry_str,
                                    int *extraction_module,
```

```
char *test_id,  
EHMS_PDataIDRecord data_id)
```

EHMS_ReadFeatureExtractionCommandTable (declaration in FEAT_features.h)

(no longer defined???)

EHMS_RedlineCheck (declaration in FEAT_features.h, definition in FEAT_features.c)

```
FEAT_Boolean EHMS_RedlineCheck (EHMS_PDataIDRecord data_id,  
float *time1, float *data1,  
float *sigmas1, float *time2,  
float *data2, float *sigmas2,  
int num_points,  
EHMS_RedlineCheckType check_type,  
EHMS_RedlineType limit_type)
```

The RedlineCheck module is used to check two PIDs to ensure that they stay within the limits defined for the given measurement. If an excursion beyond the limit defined is found, its duration is checked against a time limit value and decision logic is applied to determine if a redline has been violated.

Redline limit information is stored in the database table. All information needed for a redline check is extracted by the Redline Check module, the user need only specify (in the command table) the parameters `check_type_str` (of type `EHMS_RedlineCheckType`) and `limit_type_str` (of type `EHMS_RedlineType`). Specifying `both_pids` for `check_type_str` causes the RedlineCheck module to look for instances where both PIDs exhibit an excursion beyond the specified limit at the same time. Choosing either `pid` will indicate that only one PID must exceed the specified limit for a redline violation to occur. The last choice, `difference`, causes this module to look for redline violations in a composite data set made up of the point for point difference of the original two data sets. Note that the `both_pids` option should only be used with redundant PIDs as only one set of redline information is retrieved from the data base. Also when using the difference option a row of redline information must be present in the table `RL_INFO`, where the PID value is a string of the format `pid#- pid#`, which corresponds to the arguments for sensor and comparison sensor. The parameter `limit_type_str` specifies whether the module should look for excursions below or above the lower or upper limit. These two options are specified by the values `lower` and `upper` respectively.

Arguments:

```
data_id:  
time1:  
data1:  
sigmas1:  
time2:  
data2:  
sigmas2:  
num_points:  
check_type:  
limit_type:
```

Command table inputs:

```
expert:
```

sensor:
 sensor_postfix:
 modulename:
 starttime:
 endtime:
 compare_descrip (param1):
 check_type_str (param2):
 limit_type_str (param3):

Any features found are reported to the table F_RLVOL.

EHMS_RedundChannelChk (definition in FEAT_features.c)

```

FEAT_Boolean EHMS_RedundChannelChk (EHMS_PDataIDRecord data_id1,
                                     EHMS_PDataIDRecord data_id2,
                                     float *data1, float *data2,
                                     float *time1, float *time2,
                                     float threshold,
                                     int index_per_time_seg,
                                     int num_points1,
                                     int num_points2)

```

EHMS_RemoveLinearTrend (definition in FEAT_featureUtils.c)

```

float *EHMS_RemoveLinearTrend (float *time, float *data,
                               int num_pts)

```

EHMS_ReportPeakFeatures (declaration in FEAT_features.h, definition in FEAT_features.c)

```

void EHMS_ReportPeakFeatures (float *data, float *time,
                              float *sigmas, int fit_start,
                              int fit_end, float min_peak,
                              EHMS_PFeatureRecord feature_rec,
                              EHMS_PFeatureRecordHead list_head,
                              int num_peaks)

```

EHMS_SetThreePointWindow (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```

void EHMS_SetThreePointWindow (int counter, float *data,
                               int num_data_pts, float *start_pt,
                               float *end_pt)

```

EHMS_Smooth (definition in FEAT_featureUtils.c)

```

int EHMS_Smooth (float *pid, char *data, float *smoothed_data,
                 int num_pts, int smooth_window)

```

EHMS_TranslateTime (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
float EHMS_TranslateTime (char *test_id, char *pid,
                          char *time_str)
```

EHMS_WithinRedlineLimit (declaration in FEAT_features.h, definition in FEAT_featureUtils.c)

```
FEAT_Boolean EHMS_WithinRedlineLimit (EHMS_RedlineType limit_type,
                                       float limit,
                                       float data_point)
```

EHMS_WriteBistableResults (declaration in FEAT_features.h, definition in FEAT_fileio_t.c)

```
void EHMS_WriteBistableResults (EHMS_PDataIDRecord data_id,
                                float fit_start, float fit_end)
```

EHMS_WriteDescribeConstantThrustLevelResults (declaration in FEAT_features.h, definition in FEAT_fileio_t.c)

```
void EHMS_WriteDescribeConstantThrustLevelResults
(EHMS_PDataIDRecord data_id, float *a, float *parm_sigmas,
 float chi_square)
```

EHMS_WriteDiffertThanResults (declaration in FEAT_features.h, definition in FEAT_fileio_t.c)

```
void EHMS_WriteDiffertThanResults
(EHMS_PDifferentThanRecord test_results,
 EHMS_PDataIDRecord data_id)
```

EHMS_WriteDriftResults (definition in FEAT_fileio_t.c)

```
void EHMS_WriteDriftResults (EHMS_PDataIDRecord data_id,
                             float start_time, float end_time,
                             float start_mag, float end_mag,
                             float slope)
```

EHMS_WriteErraticResults (declaration in FEAT_features.h, definition in FEAT_fileio_t.c)

```
void EHMS_WriteErraticResults (EHMS_PDataIDRecord data_id,
                               float start, float end)
```

EHMS_WriteFeatureRecord (declaration in FEAT_features.h, definition in FEAT_fileio_t.c)

```
void EHMS_WriteFeatureRecord (EHMS_PFeatureRecord feature_rec,
                              EHMS_PDataIDRecord data_id)
```

EHMS_WriteIsFlatResults (declaration in FEAT_features.h, definition in FEAT_fileio_t.c)

```
void EHMS_WriteIsFlatResults (EHMS_PDataIDRecord data_id,  
                             float slope)
```

EHMS_WriteLevelShiftResults (declaration in FEAT_features.h, definition in FEAT_fileio_t.c)

```
void EHMS_WriteLevelShiftResults (EHMS_PDataIDRecord data_id,  
                                  float start_time,  
                                  float end_time, float last_mag,  
                                  float new_mag)
```

EHMS_WriteNoiseResults (definition in FEAT_fileio_t.c)

```
void EHMS_WriteNoiseResults (EHMS_PDataIDRecord data_id,  
                             float start_time, float end_time)
```

EHMS_WriteRedlineViolationRecord (definition in FEAT_fileio_t.c)

```
void EHMS_WriteRedlineViolationRecord  
(EHMS_PDataIDRecord data_id, float start_time, float end_time,  
 EHMS_RedlineCheckType check_type, EHMS_RedlineType limit_type,  
 float limit, char *offending_pid)
```

EHMS_WriteRedundResults (definition in FEAT_fileio_t.c)

```
void EHMS_WriteRedundResults (EHMS_PDataIDRecord data_id,  
                              EHMS_PDataIDRecord data_id2,  
                              float start, float end)
```

EHMS_WriteSpikeResults (declaration in FEAT_features.h, definition in FEAT_fileio_t.c)

```
void EHMS_WriteSpikeResults (EHMS_PDataIDRecord data_id,  
                             float start_time, float end_time,  
                             float magnitude)
```

EHMS_WriteTheFeatureRecords (declaration in FEAT_features.h, definition in FEAT_fileio_t.c)

```
void  
EHMS_WriteTheFeatureRecords (int peak_set,  
                             EHMS_PFeatureRecordHead list_head,  
                             EHMS_PDataIDRecord data_id)
```

EHMS_ZeroShiftCheck (declaration in FEAT_features.h, definition in FEAT_features.c)

```
FEAT_Boolean EHMS_ZeroShiftCheck (EHMS_PDataIDRecord data_id,  
                                   float *time, float *data,  
                                   int num_data_pts,  
                                   float low_limit,  
                                   float upper_limit)
```

FEAT_AlarmHandler (declaration in **FEAT_features.h**, definition in **FEAT_features.c**)

```
void FEAT_AlarmHandler()
```

FEAT_BuildIncludeList (definition in **FEAT_featureUtils.c**)

```
int FEAT_BuildIncludeList (char ***include_list)
```

FEAT_GetPostFixSSMEData (declaration in **FEAT_features.h**, definition in **FEAT_featureUtils.c**)

```
int FEAT_GetPostFixSSMEData (char *test, char *pfxpid,  
                             float start, float stop,  
                             struct SSME_data **SSMEData)
```

features_exec (definition in **FEAT_features.c**)

```
features_exec (char *test_id)
```

The main driver of the feature extractor is responsible for building the array of run time commands based on the contents of the command table **F_COMM**, found in the **SSME_DB** database, and the results of a thrust profile analysis. Each entry in the run time command array represents a call to a specific feature extraction module with parameters determined by the results of the thrust profile analysis for the test of interest and the contents of the command table. The main driver loops through this array of commands executing each named module with the specified parameters. The results of each feature search are appended to a temporary output file, found in **/tmp**, where a separate file is created for each type of feature. After all commands in the run time command array have been executed, the contents of these temporary files are loaded into the corresponding Ingres feature tables and the files deleted.

Thrust profile analysis of a test consists of detecting all periods of constant thrust over the entire range of data stored for **EHMS_ThrustPid**, which is currently defined to be the one-second average of **PID 63** (**PID "63A"**). **PID 63** represents the engine's response to the commanded throttle value. Commanded throttle was originally chosen for this analysis as it is noise free, making thrust level changes easy to detect and the percent throttle value easy to determine. Currently, response to commanded throttle is used. It is less prone to falsely indicate as steady state, periods of changing thrust. It was found that in some cases the rise time of the engine to a throttle step was significant enough to cause the feature extractor to flag transient effects as features. In the interest of cutting down on the amount of features to be analyzed by the preprocessor (**HFILTER**) the **PID** used for thrust level analysis was changed from **PID "287a"** to **PID "63a"**.

Each period of constant thrust detected has its start time adjusted to account for settling time, such that the resulting interval represents 95 percent of the original time segment (with a minimum allowance of one second and a maximum of three seconds). This operation moves the end points of the interval away from thrust transition times. This helps eliminate the effect of transients which may manifest themselves as features.

For each entry in the command table, where **STARTTIME** equals **ENDTIME**, a corresponding entry is made in the run time command array for each period of constant thrust. In this way only features which occur during periods of constant thrust are recorded, with each feature tagged by the thrust level at which it occurred. The user can override this operation and set predetermined start and end times for a feature search in a specific measurement, however if this time period spans more than one thrust level, any features found will carry thrust level ids of -999.

Thrust profile analysis is made automatically for each test. Each period of constant thrust is classified and stored in the database table F_THLEDE.

The feature extractor may operate with either one second average or full sample rate data. Because many features happen over time scales of several seconds or longer, it is appropriate to fit them using one second average data. For each one second time bin there exists a standard deviation. These values are used to help track the movement of a measurement and provide the ability to correctly account for the noise in the data. Full sample data is used when necessary for determination of high speed features such as the 1/3 - 1/2 Hz oscillation in PBP bistability. When full sample data is used, only short intervals should be extracted from the data base to minimize memory use and maximize execution speed.

In the (hopefully rare) event of a severe run time error (such as the attempt to solve a singular system of equations) during the execution of any given command, the feature extractor main driver will log the complete text of the current command string in the file BAD_COMMANDS. This file will be found in the users current working directory. If BAD_COMMANDS already exists in the users current working directory, the command which produced the error will be appended at the end of the file. Note that the feature extractor does not halt execution in the event of a run time error. Any command which produces a severe run time error during a calculation, from which there can be no graceful recovery, causes a Unix signal 14 (SIGALRM). The handler routine assigned to SIGALRM, logs the current command in BAD_COMMANDS and executes a longjmp to cause feature extraction to resume with the next command found in the command table.

FindMIAPids (definition in FEAT_dbutils.c)

```
int FindMIAPids (char **include_list, int num_included_pids,
                char *test_id)
```

fit (definition in FEAT_featurefits.c)

```
fit (float * y, float *x, int n, int ms, int mf, double *c,
     float *sigmas, double *chi_square, double *fit_std_dev)
```

gaussj (declaration in FEAT_featurefits.h, definition in FEAT_featurefits.c)

```
void gaussj (float **a, int n, float **b, int m)
```

kstwo (declaration in FEAT_featurefits.h, definition in FEAT_featurefits.c)

```
void kstwo (float data1[], int n1, float data2[], int n2,
            float *d, float *prob)
```

ludcmq (definition in FEAT_featurefits.c)

```
ludcmq (double **a, int n, int ndim)
```

moment (definition in FEAT_featurefits.c)

```
void moment (float data[], int n, float *ave, float *adev,
             float *sdev, float *svar, float *skew, float *curt)
```

mrqcof (declaration in FEAT_featurefits.h, definition in FEAT_featurefits.c)

```
void mrqcof (float x[], float y[], float sig[], int ndata,
             float a[], int ma, int lista[], int mfit,
```

```
float **alpha, float beta[], float *chisq,
void (*funcs)(), float *fit_std_dev)
```

mrqmin (declaration in FEAT_featurefits.h, definition in FEAT_featurefits.c)

```
void mrqmin (float x[], float y[], float sig[], int ndata, int a,
            int ma, int lista[], int mfit, float **covar,
            float **alpha, float *chisq, void (*funcs)(),
            float *alamda, float *fit_std_dev)
```

Source code for this routine, as well as a thorough discussion of its operation, can be found in section 14.4 of *Numerical Recipes in C, The Art of Scientific Computing* (Cambridge University Press, 1988).

probks (declaration in FEAT_featurefits.h, definition in FEAT_featurefits.c)

```
float probks (float alam)
```

solnq (definition in FEAT_featurefits.c)

```
solnq (double **a, double *b, int n, int ndim)
```

sort (declaration in FEAT_featurefits.h, definition in FEAT_featurefits.c)

```
void sort (int n, float ra[])
```

svbksb (definition in FEAT_featurefits.c)

```
void svbksb (float **u, float w[], float **v, int m, int n,
            float b[], float x[])
```

svdcmp (definition in FEAT_featurefits.c)

```
void svdcmp (float **a, int m, int n, float *w, float **v)
```

svdfit (definition in FEAT_featurefits.c)

```
void svdfit (float x[], float y[], float sig[], int ndata,
            float a[], int ma, float **u, float **v,
            float w[], float *chisq, void (*funcs)(),
            float *fit_std_dev)
```

svdvar (definition in FEAT_featurefits.c)

```
void svdvar (float **v, int ma, float w[], float **cvm)
```

WriteMissingPids (definition in FEAT_dbutils_t.c)

```
void WriteMissingPids (char *test_id, char *pid)
```

WriteZeroShiftCheckResults (definition in FEAT_fileio_t.c)

```
void WriteZeroShiftCheckResults (EHMS_PDataIDRecord data_id,
                                float average, float offset)
```

5.7 Algorithm Comments

5.7.1 General Comments

For the collection of certain classes of features, limiting feature extraction to periods of constant thrust is inadequate to insure collection of valid features. Due to the methods employed in the feature extraction routines: FindErraticBehaviour, FindLevelShift, DeltaLevelShift and FindSpike, the effects of fuel tank repressurization must be accounted for. The feature classes noted above should only be collected during overlapping periods of constant thrust level and linear fuel tank repressurization/venting behavior.

Evidence of fuel tank repressurization and venting are searched for in one second average data for pid labeled, 'LPOTP Pump Discharge Pressure A' (nominally PID #209). There is no direct measurement of this behavior so 'LPOTP Pump Discharge Pressure A' is used as it was deemed the most sympathetic to the effects of interest. Repressurization and venting are indicated in PID 209 as linear periods which deviate from the usual horizontal trace, giving sections of the sensor data a sort of saw tooth effect. The function which detects these periods, EHMS_FindLinearLPOTPDDischargePressure, takes advantage of the observation that all sections of the trace, including those where venting or repressurization is taking place, exhibit a constant slope. The recursive routine EHMS_FindInflectionPtsByLinearTrendRemoval is called by EHMS_FindLinearLPOTPDDischargePressure to locate the start and end points of all periods of linear behavior in PID 209. By searching for features in other traces only during these linear periods (as well as periods of constant thrust) a reasonable assumption of steady state behavior can be made. The effects of venting and repressurization on other sympathetic pids are also negated and no longer cause erroneous periods of erratic behavior to be detected.

While EHMS_FindLinearLPOTPDDischargePressure does break out any periods displaying venting or repressurization effects, it makes no judgment about the data it generates. The algorithm was designed to negate the effects of venting and repressurization for the purposes of feature extraction. The feature extraction program does not need to know if the effects are occurring, it only needs to account for them.

Another source of potential error in feature extraction is produced by "bit_toggle". This effect is introduced by measuring a physical quantity with a sensor having limited dynamic range. The quantization error introduced into the measurements by using a quantizing step size as large as deviations of interest, produces a toggling effect seen in a graph of the collected data. In order to account for this type of error (which most often manifests itself as erroneous data spikes) the feature extractor searches for the smallest non-trivial step difference between two contiguous data points on the given interval. This value is taken to be the step size used by the sensor to quantize the analog measurements. By adjusting a calculated parameter up or down by half the step size, we may account for bit toggle error. One half the step size is equivalent to the maximum error in any given measurement. This toggle extraction technique is used by the following feature extractor functions: EHMS_FindErraticBehaviour, EHMS_DeltaDifferentThan, and EHMS_FindSpike.

5.7.2 DifferentThan Module

A first order polynomial fit is made to a composite data set where the data points and standard deviations are given by

```
delta_data[i] = data1[i] - data2[i]
```

```
delta_sigmas[i] = sqrt((double) (sigmas1[i] * sigmas1[i] +
```

sigmas2[i] * sigmas2[i]))

A fit which produces a line with little slope indicates that the two curves track each other well. The constant offset term produced by the fit indicates the distance maintained between the two data sets. A small value within acceptable error limits indicates that the two curves may be drawn from the same parent population. A constant offset term which exceeds acceptable error limits is indicative of a constant offset maintained between two curves. For the case where both fitted coefficients are outside error limits the two curves are determined to be "different".

After the fit is made to the composite data set, a probability measurement is made to determine if the two data set were drawn from the same parent population. This measurement is based on the Kolmogorov-Smirnov test. Small values of the measurement indicate that the cumulative distribution function of the first data set is significantly different from the second. This value, as well as the maximum step size found in either data set and the maximum average magnitude found on either data set, is used in the logic described below to determine if the two data sets can be considered to be "the same" or to differ by a constant offset.

If the result of the Kolmogorov-Smirnov test is greater than the value defined by EHMS_SameAsProbability, or three times the standard deviation calculated for the fit is less than half the maximum average magnitude, then a further check is made to determine if the higher order coefficients differ with acceptable error. If for each fitted parameter 1 through N (where parm_sigmas is the sigma for the corresponding parameter, adjusted to account for bit toggle)

```
param[i] + (num_comparison_sigmas * parm_sigmas[i]) > 0.0
and
param[i] - (num_comparison_sigmas * parm_sigmas[i]) < 0.0
```

then the two data sets are judged to be drawn from the same parent population. If the above is true for all parameters except the constant term, then the two data sets are said to differ by a constant offset. The offset value is given by param[1], and the standard deviation for this value is given by parm_sigmas[1]. If none of the above is true and, all parameters 1 through N have statistically significant terms, then the two data sets are considered to be drawn from different parent populations.

The two most common uses of the DifferentThan module are listed below, along with a short description of how to interpret the results produced.

1. Checking that redundant sensors are tracking each other within statistical limits.

To make this type of check, add an entry to the table F_COMM containing the command table inputs listed above. Setting STARTTIME and ENDTIME equal will cause the sensors to be compared for all periods of constant thrust. After the feature extractor has run, select the different than records for the current test (TESTID) and the first PID listed in the command (sensor) from the table F_COMM. Inspect the COEF_W_ERR_B field of each record. If any records are found which have a value of "False" for this field then the two sensor traces compared were found to differ statistically from each other. More specifically, the coefficients of the straight line fit to the difference data set were not within the tolerance set by num_comparison_sigmas. Further verification of the probability that the two data sets are not drawn from the same parent population can be determined by inspection of the field CHI_SQUARE which represents the goodness of the fit to the difference data set and PROB which is a measure of the actual probability that the two data sets were drawn from the same parent population. Small values of PROB indicate that the two data sets are significantly different.

2. Checking for sensors which differ by a constant offset.

This check is handled in the same manner as the check described above but involves inspection of the field DIFF_BY_OFFSE. If a value of "True" is found in this field, then the two data sets were found to track each other but with a constant offset. The value of this offset is found in the field OFFSET. The sign on this value is determined by the convention used to compute the difference data set used for the fit

$$\text{delta_data}[i] = \text{data1}[i] - \text{data2}[i]$$

A positive value indicates that the first data set listed in the table (data1) has values greater than the second (data2) by a constant term listed in the table field offset.

5.8 Cflow output

```

1  main: int(), <FEAT_main.c 75>
7  features_exec: int(), <FEAT_features.c 4586>
10  init_PTDS_tekbase: <>
11  FEAT_BuildIncludeList: int(), <FEAT_featureUtils.c 41>
22  FindMIAPids: int(), <FEAT_dbutils_t.c 653>
23  DBIO_GetPidInfo: void(), <FEAT_dbutils_t.c 217>
26  DATA_info: <>
27  STRNG_RemoveTrailingSpaces: char*(), <STRNG_utils.c 76>
31  WriteMissingPids: void(), <FEAT_dbutils_t.c 641>
33  tbl_put: <>
35  DBIO_GetTheFeatureExtractionCommands: char**(), <FEAT_dbutils_t.c 484>
37  tbl_count: <>
41  tbl_get: <>
42  STRNG_RemoveTrailingSpaces: 27
46  EHMS_AnalyzeThrustProfile: struct*(), <FEAT_features.c 3701>
50  EHMS_CreateDataIDRecord: struct*(), <FEAT_featureUtils.c 1336>
54  DBIO_GetPidInfo: 23
55  FEAT_GetPostFixSSMEData: int(), <FEAT_featureUtils.c 1927>
58  DATA_average: <>
59  DATA_get: <>
63  EHMS_FindConstantThrust: struct*(), <FEAT_features.c 3309>
64  EHMS_SetThreePointWindow: void(), <FEAT_featureUtils.c 953>
65  EHMS_GetFitInterval: void(), <FEAT_featureUtils.c 786>
67  EHMS_CreateDataIDRecord: 50
68  EHMS_CopyDataID: void(), <FEAT_featureUtils.c 676>
70  EHMS_DescribeConstantThrustLevel: int(), <FEAT_features.c 1522>
71  EHMS_GetNumberBasisFuncs: int(), <FEAT_featurefits.c 312>
72  EHMS_Gaussian: void(), <FEAT_featurefits.c 136>
74  EHMS_FastRiseExpFall: void(), <FEAT_featurefits.c 100>
76  EHMS_NthOrderPoly: void(), <FEAT_featurefits.c 65>
79  EHMS_NthOrderPoly: 76
82  EHMS_GetMRQMemory: void(), <FEAT_featureUtils.c 1548>
85  EHMS_MakeFit: void(), <FEAT_featurefits.c 188>
86  mrqmin: void(), <FEAT_featurefits.c 235>
87  matrix: float**(), <nrutil.c 65>
89  nrerror: void(), <nrutil.c 13>
92  alarm: <>
94  vector: float*(), <nrutil.c 33>
96  nrerror: 89
97  nrerror: 89
98  mrqcof: void(), <FEAT_featurefits.c 912>
99  vector: 94
101  free_vector: void(), <nrutil.c 139>
103  gaussj: void(), <FEAT_featurefits.c 832>
104  ivector: int*(), <nrutil.c 43>

```

```

106         nrerror: 89
109         nrerror: 89
110         free_ivector: void(), <nrutil.c 145>
112         covsrt: void(), <FEAT_featurefits.c 965>
113         free_vector: 101
114         free_matrix: void(), <nrutil.c 161>
117         EHMS_WriteDescribeConstantThrustLevelResults: void(),
                                     <FEAT_fileio_t.c 127>

119         tbl_put: 33
121         EHMS_FreeMRQMemory: void(), <FEAT_featureUtils.c 1608>
124         EHMS_AddThrustPeriodID: void(), <FEAT_featureUtils.c 496>
130         DATA_release: <>
131         EHMS_BuildRunTimeCommandArray: char**(), <FEAT_featureUtils.c 312>
135         DBA_StandardStringToPid: void(), <FEAT_dbutils_t.c 59>
138         tbl_get: 41
140         STRNG_RemoveTrailingSpaces: 27
141         EHMS_MatchPostfix: void(), <FEAT_featureUtils.c 111>
143         EHMS_TranslateTime: float(), <FEAT_featureUtils.c 255>
145         EHMS_GetEngineCutoff: float(), <FEAT_dbutils_t.c 345>
147         tbl_get: 41
149         DBIO_GetPidInfo: 23
152         DBIO_GetPidInfo: 23
158         EHMS_GetThrustLevel: float(), <FEAT_featureUtils.c 150>
162         signal: <>
163         FEAT_AlarmHandler: void(), <FEAT_features.c 4504>
164         longjmp: <>
166         setjmp: <>
172         EHMS_FeatureExtractor: enum(), <FEAT_features.c 3795>
173         EHMS_CreateDataIDRecord: 50
174         DBIO_GetPidInfo: 23
175         EHMS_FindLinearLPOTPDDischargePressure: struct*(),
                                     <FEAT_featureUtils.c 2134>

176         DBA_TimeRangeToRecNumRange: void(), <FEAT_featureUtils.c 1020>
177         DBA_StandardStringToPid: 135
178         EHMS_MatchPostfix: 141
179         FEAT_GetPostFixSSMEDData: 55
180         EHMS_FindInflectionPtsByLinearTrendRemoval: void(),
                                     <FEAT_featureUtils.c 2052>

182         EHMS_RemoveLinearTrend: float*(), <FEAT_featureUtils.c 1973>
184         EHMS_FindInflectionPtsByLinearTrendRemoval: 180
185         EHMS_AddLinearPressureRec: void(), <FEAT_featureUtils.c 2010>
190         EHMS_AddLinearPressureRec: 185
191         DATA_release: 130
192         EHMS_ParseTableEntry: enum(), <FEAT_featureUtils.c 1055>
195         DBIO_GetPidInfo: 23
197         FEAT_GetPostFixSSMEDData: 55
199         cfree: <>
201         DATA_info: 26
202         EHMS_GetEngineCutoff: 145
204         EHMS_Smooth: int(), <FEAT_featureUtils.c 1646>
208         EHMS_GetSmoothedOffset: int(), <FEAT_featureUtils.c 1689>
209         EHMS_MaxMin: int(), <FEAT_featureUtils.c 1715>
210         EHMS_ArrayMult: int(), <FEAT_featureUtils.c 1749>
211         DATA_release: 130
212         EHMS_FindStepSize: float(), <FEAT_featureUtils.c 2211>
213         DBIO_GetConstantThrustLevels: void(), <FEAT_dbutils_t.c 287>
216         tbl_count: 37
220         tbl_get: 41

```



```

224         DBA_TimeRangeToRecNumRange: 176
225         FEAT_GetPostFixSSMEData: 55
227         DATA_release: 130
229     EHMS_RedundChannelChk: enum(), <FEAT_features.c 634>
231         EHMS_WriteRedundResults: void(), <FEAT_fileio_t.c 790>
233         tbl_put: 33
236     EHMS_NoisyPid: enum(), <FEAT_features.c 808>
237         EHMS_WriteNoiseResults: void(), <FEAT_fileio_t.c 217>
239         tbl_put: 33
240         DBIO_PidToSensorLabel: char*(), <FEAT_dbutils_t.c 112>
242         tbl_get: 41
245     EHMS_ZeroShiftCheck: enum(), <FEAT_features.c 4536>
246         WriteZeroShiftCheckResults: void(), <FEAT_fileio_t.c 172>
248         tbl_put: 33
249         DBIO_PidToSensorLabel: 240
251     EHMS_FindDrift: enum(), <FEAT_features.c 1259>
253         EHMS_WriteDriftResults: void(), <FEAT_fileio_t.c 267>
255         tbl_put: 33
256         DBIO_PidToSensorLabel: 240
258     EHMS_IsFlat: enum(), <FEAT_features.c 1413>
259         EHMS_GetNumberBasisFuncs: 71
260         EHMS_NthOrderPoly: 76
263         EHMS_GetMRQMemory: 82
264         EHMS_MakeFit: 85
266         EHMS_WriteIsFlatResults: void(), <FEAT_fileio_t.c 327>
268         tbl_put: 33
269         DBIO_PidToSensorLabel: 240
271         EHMS_FreeMRQMemory: 121
273     EHMS_FindSpike_b: enum(), <FEAT_features.c 2864>
274         EHMS_GetNumberBasisFuncs: 71
275         EHMS_NthOrderPoly: 76
276         EHMS_GetMRQMemory: 82
277         calc_stddev: double(), <FEAT_featureUtils.c 2314>
279         EHMS_MakeFit: 85
280         EHMS_FreeMRQMemory: 121
283         EHMS_WriteSpikeResults: void(), <FEAT_fileio_t.c 422>
285         tbl_put: 33
286         DBIO_PidToSensorLabel: 240
288     EHMS_DetectBistability: enum(), <FEAT_features.c 3557>
289         EHMS_GetFitIntervalForSpikeCheck: void(), <FEAT_featureUtils.c 904>
290         FEAT_GetPostFixSSMEData: 55
292         EHMS_MakeDummySigmas: float*(), <FEAT_featureUtils.c 984>
295         EHMS_GetMRQMemory: 82
296         EHMS_MakeFit: 85
297         EHMS_NthOrderPoly: 76
298         EHMS_FreeMRQMemory: 121
299         EHMS_WriteBistableResults: void(), <FEAT_fileio_t.c 529>
301         tbl_put: 33
302         DBIO_PidToSensorLabel: 240
305         DATA_release: 130
306     EHMS_FindLevelShift: enum(), <FEAT_features.c 2357>
308         EHMS_GetNumberBasisFuncs: 71
309         EHMS_NthOrderPoly: 76
310         EHMS_CreateConstantPeriodRecord: struct*(),
311             <FEAT_featureUtils.c 1374>
313     EHMS_DescribeConstantPeriod: void(), <FEAT_features.c 1632>
314         EHMS_GetNumberBasisFuncs: 71
315         EHMS_NthOrderPoly: 76

```

```

318         EHMS_GetMRQMemory: 82
319         EHMS_MakeFit: 85
321         EHMS_FreeMRQMemory: 121
323         EHMS_AddConstantPeriodID: void(), <FEAT_featureUtils.c 542>
330         EHMS_WriteLevelShiftResults: void(), <FEAT_fileio_t.c 474>
332         tbl_put: 33
333         DBIO_PidToSensorLabel: 240
335     EHMS_FindErraticBehaviour: enum(), <FEAT_features.c 1724>
336         EHMS_GetNumberBasisFuncs: 71
337         EHMS_NthOrderPoly: 76
338         EHMS_GetMRQMemory: 82
339         EHMS_MakeFit: 85
340         EHMS_WriteErraticResults: void(), <FEAT_fileio_t.c 376>
342         tbl_put: 33
343         DBIO_PidToSensorLabel: 240
346         EHMS_FreeMRQMemory: 121
348     EHMS_FindPeak: enum(), <FEAT_features.c 476>
349         EHMS_CreateFeatureRecordHead: struct*(), <FEAT_featureUtils.c 1455>
352         EHMS_SetThreePointWindow: 64
353         EHMS_CreateFeatureRecord: struct*(), <FEAT_featureUtils.c 1427>
357         EHMS_ReportPeakFeatures: void(), <FEAT_features.c 271>
360         EHMS_Gaussian: 72
361         EHMS_FastRiseExpFall: 74
362         EHMS_GetNumberBasisFuncs: 71
363         EHMS_GetMRQMemory: 82
364         EHMS_MakeFit: 85
365         EHMS_FreeMRQMemory: 121
366         EHMS_AddFeatureRecord: void(), <FEAT_featureUtils.c 1262>
367         EHMS_WriteTheFeatureRecords: void(), <FEAT_fileio_t.c 738>
368         EHMS_WriteFeatureRecord: void(), <FEAT_fileio_t.c 676>
370         tbl_put: 33
371         DBIO_PidToSensorLabel: 240
373         EHMS_FreePeakRecords: void(), <FEAT_featureUtils.c 1486>
375         EHMS_DeleteFeatureRecord: void(), <FEAT_featureUtils.c 1296>
377     DBA_StandardStringToPid: 135
378     EHMS_MatchPostfix: 141
379     EHMS_MakeDummySigmas: 292
380     EHMS_RedlineCheck: enum(), <FEAT_features.c 1863>
382         DBIO_FetchRedlineInfo: void(), <FEAT_dbutils_t.c 380>
385         tbl_count: 37
389         tbl_get: 41
391         index: <>
393         EHMS_GetCompositeDataSet: float**(), <FEAT_featureUtils.c 585>
397         EHMS_WithinRedlineLimit: enum(), <FEAT_featureUtils.c 643>
398         EHMS_WriteRedlineViolationRecord: void(), <FEAT_fileio_t.c 58>
400         tbl_put: 33
401         DBIO_PidToSensorLabel: 240
405     WPREV_GetPreviousTest: int(), <WPREV_findtest.c 43>
406         WPREV_GetTests: void(), <WPREV_db_t.c 44>
408         tbl_count: 37
411         tbl_get: 41
413         STRNG_RemoveTrailingSpaces: 27
416     WPREV_GetThrustDescriptions: void(), <WPREV_db_t.c 156>
418         tbl_count: 37
421         tbl_get: 41
422         STRNG_RemoveTrailingSpaces: 27
424     WPREV_GetAnomalies: void(), <WPREV_db_t.c 111>
426         tbl_count: 37

```

```

429         tbl_get: 41
430         STRNG_RemoveTrailingSpaces: 27
435         WPREV_FreeLists: void(), <WPREV_findtest.c 376>
439         EHMS_GetThrustLevelIntersection: enum(), <FEAT_featureUtils.c 186>
440         DBIO_GetConstantThrustLevels: 213
442         EHMS_CreatedDifferentThanRecord: struct*(), <FEAT_featureUtils.c 1401>
445         EHMS_DeltaDifferentThan: void(), <FEAT_features.c 2113>
449         EHMS_GetNumberBasisFuncs: 71
450         EHMS_NthOrderPoly: 76
451         EHMS_GetMRQMemory: 82
452         EHMS_MakeFit: 85
453         EHMS_FreeMRQMemory: 121
454         kstwo: void(), <FEAT_featurefits.c 706>
455         sort: void(), <FEAT_featurefits.c 745>
457         probks: float(), <FEAT_featurefits.c 796>
461         EHMS_FindStepSize: 212
463         EHMS_WriteDiffertThanResults: void(), <FEAT_fileio_t.c 576>
466         tbl_put: 33
467         DBIO_PidToSensorLabel: 240
470         EHMS_DeltaLevelShift: enum(), <FEAT_features.c 2049>
471         EHMS_GetCompositeDataSet: 393
474         EHMS_FindLevelShift: 306
476         EHMS_BalancePistonCheckInit: enum(), <FEAT_features.c 79>
477         DBA_StandardStringToPid: 135
479         EHMS_MatchPostfix: 141
480         DBIO_GetPidInfo: 23
481         FEAT_GetPostFixSSMEData: 55
482         EHMS_GetThrustLevelIntersection: 439
483         EHMS_GetCompositeDataSet: 393
486         DATA_release: 130
487         RSRC_UpdateResource: void(), <RSRC_rlist.c 125>
488         RSRC_GetResourceList: void(), <RSRC_dbutils_t.c 211>
490         tbl_count: 37
493         tbl_get: 41
494         STRNG_RemoveTrailingSpaces: 27
496         RSRC_FindResourceId: int(), <RSRC_rlist.c 87>
498         RSRC_UpdateResourceBoardResourceValue: void(), <RSRC_dbutils_t.c 84>
501         tbl_get: 41
504         RSRC_GetBits: int(), <RSRC_rlist.c 240>
505         tbl_update: <>
507         RSRC_FreeResourceList: void(), <RSRC_rlist.c 158>
508         cfree: 199
510         tbl_free_all: <>

```

6. External Effects Program (external)

6.1 Source Files

`external.c`: The main program for the External Effects Module and many of the subroutines written specifically for this module.

`dbAccess.c`: The functions which access the TekBase databases for retrieving data.

`loadData.c`: Functions to allocate memory and load data for a specified test database and for PIDs and to free the memory allocated.

`removeEffect.c`: The functions which perform the calculations given in the Algorithm section above for removing the external effects.

`smoothData.c`: Function to smooth the data for a PID by averaging over a specified interval, in effect, a simple convolution filter.

`writeNPID.c`: Functions to generate the output file.

6.2 Header Files

`external.h`: Header file containing the declarations required for `external.c`.

`dbAccess.h`: Header file containing the type definitions and declarations of the functions defined in `dbAccess.c`.

`loadData.h`: Header file containing the declarations of the functions defined in `loadData.c`.

`removeEffect.h`: Header file containing the declarations of the functions defined in `removeEffect.c`.

`smoothData.h`: Header file containing the declaration of the function defined in `smoothData.c`.

`writeNPID.h`: Header file containing the declaration of the function defined in `writeNPID.c` which is available to other functions.

6.3 Defined Constants

`DB_SERVER`: Default TekBase database server to use if the environment variable `QYHOST` is not set; value "jetson"; defined in `dbAccess.h`.

`FALSE`: Used as a flag value when delta PIDs are not to be normalized and as the return value if failure occurs in some functions, including `dbGetCompInfo` and `writeNPID`; value 0; defined in `external.h`.

`INDEPENDENT_PIDS`: Number of independent PIDs used in removing external effects; value 6; defined in `loadData.h`.

`MAX_FILES`: Maximum number of data files that can be associated with a test; value 9; defined in `loadData.h`.

`SMOOTH_WINDOW`: Width of window used for averaging data when smoothing raw PID data, currently not utilized; value 51; defined in `smoothData.h`.

`SMOOTH_WIN_D`: Width of the window used for averaging data when smoothing the unnormalized delta between the dependent PIDs for the current and comparison test; value 51; defined in `smoothData.h`.

`SMOOTH_WIN_I`: Width of the window used for averaging data when smoothing the deltas between the independent PIDs for the current and comparison test; value 51; defined in `smoothData.h`.

`TRUE`: Used as a flag value when delta PIDs are to be normalized and as the return value by some functions, including `dbGetCompInfo` and `writeNPID`, if successful; value 1; defined in `external.h`.

6.4 Defined Types

`CompDesc` (defined in `dbAccess.h`)

```
typedef struct {  
    TSTRING
```

```

        comp_test_id[13];
    } CompDesc;

```

This structure is used in retrieving the comparison test corresponding to the current test from the CMP_DESC table of the TekBase database SSME_DB.

Members:

comp_test_id: Test ID for the comparison test.

DataBase (defined in loadData.h)

```

typedef struct {
    int
        num_file;
    char
        db_name[40],
        *file_name[MAX_FILES];
    fileCtlData
        *file[MAX_FILES];
    pidInfo
        *pid[INDEPENDENT_PIDS];
} DataBase;

```

This structure contains the information required by the program for the data files and independent PIDs for a test.

Members:

num_file: Number of test data files associated with the test.

db_name: Name of the test. This may be the name of a single test data file or the test name taken from the first line of the database directory file for the test, depending on the value of CureentTest used in the command line to run the external effects module.

file_name: Array of file names for all test data files associated with the test.

file: File descriptor and other database information for each file associated with the test.

pid: Information and data for each independent PID for the test.

External_Effects (defined in dbAccess.h)

```

typedef struct {
    TSTRING
        description[41];
    TREAL
        PL,
        MR,
        H2P,
        O2P,
        H2T,
        O2T,
        LSQCOFF1,
        LSQCOFF2,
        LSQCOFF3,
        LSQCOFF4,
        LSQCOFF5,

```

```

        LSQCOFF6;
    } External_Effects;

```

This structure is used in retrieving the gains table information required by the external effects module for each of the dependent PIDs from the GAINS table of the TekBase database EXTERNAL.

Members:

description: Description of the dependent PID.

PL: Linear gain associated with the Power Level (not used because the power level effect is assumed to be nonlinear).

MR: Linear gain associated with the Mixture Ratio.

H2P: Linear gain associated with the LPFP Inlet Pressure.

O2P: Linear gain associated with the LPOP Inlet Pressure.

H2T: Linear gain associated with the LPFP Inlet Temperature.

O2T: Linear gain associated with the LPOP Inlet Temperature.

LSQCOFF1: Constant term of the fifth degree polynomial approximating the power level effects.

LSQCOFF2: Coefficient of the first degree term of the fifth order polynomial approximating the power level effects.

LSQCOFF3: Coefficient of the second degree term of the fifth order polynomial approximating the power level effects.

LSQCOFF4: Coefficient of the third degree term of the fifth order polynomial approximating the power level effects.

LSQCOFF5: Coefficient of the fourth degree term of the fifth order polynomial approximating the power level effects.

LSQCOFF6: Coefficient of the fifth degree term of the fifth order polynomial approximating the power level effects.

fileCtlData (defined in loadData.h)

```

typedef struct {
    char
        test_title[41];
    float
        cutoff;
    struct file_control_data
        *fileData;
} fileCtlData;

```

This structure contains information used by the external effects module concerning a test data file.

Members:

test_title: Title for the test.

cutoff: Cutoff time for the test.

fileData: Additional file information used by the dbacc_v2 library for accessing the data file (See dbacc_v2.h for the definition of the structure file_control_data).

gainsTbl (defined in external.h)

```

typedef struct {
    char

```



```

        description[41];
double
    PL,
    MR,
    H2P,
    O2P,
    H2T,
    O2T,
    lsqcoef[6];
} gainsTbl;

```

This structure contains a description of a dependent PID and the linear gains and least-squares coefficients used in removing the effects of the independent PIDs for that dependent PID.

Members:

description: Description of the dependent PID.

PL: Linear gain associated with the Power Level (not used because the power level effect is assumed to be nonlinear).

MR: Linear gain associated with the Mixture Ratio (not used because the program currently does not remove the effect of the mixture ratio).

H2P: Linear gain associated with the LPFP Inlet Pressure.

O2P: Linear gain associated with the LPOP Inlet Pressure.

H2T: Linear gain associated with the LPFP Inlet Temperature.

O2T: Linear gain associated with the LPOP Inlet Temperature.

lsqcoef: Least-squares coefficients of the fifth degree polynomial approximating the power level effects.

hardwareDeltaInfo (defined in dbAccess.h)

```

typedef struct {
    TSTRING
        test_id[13],
        parameter[61];
    TREAL
        start_delta;
} hardwareDeltaInfo;

```

This structure is used in retrieving the constant delta due to changes in the hardware configuration between the current and comparison tests for each of the dependent PIDs from the DELTAS table of the TekBase database SSME_DB.

Members:

test_id: Test ID for the current test.

parameter: Description of the dependent PID for which the hardware delta is requested.

start_delta: The constant delta for the dependent PID due to changes in the hardware configuration between the current and comparison tests.

LowRate (defined in external.h)

```

typedef struct {
    float

```

```

        rate;
    long int
        points;
    float
        *time;
} LowRate;

```

This structure contains information concerning the lowest sampling rate among the independent PIDs for both the current and comparison tests.

Members:

rate: Lowest sampling rate among the independent PIDs for both the current and comparison tests.

points: Number of data points for the PID with the lowest sampling rate.

time: Pointer into time array for the test corresponding to the beginning time of the PID with the lowest sampling rate.

pidInfo (defined in `loadData.h`)

```

typedef struct {
    char
        *name,
        *units,
        *descr;
    int
        npa,
        points;
    float
        rate,
        t_start,
        t_stop,
        *data,
        *time;
} pidInfo;

```

This structure is used to store the PID information required by the external effects module.

Members:

name: Name of the PID.

units: Units for the PID data.

descr: Description of the PID.

npa: Number of points available for the PID.

points: Actual number of points stored for the PID.

rate: Sampling rate for the PID data.

t_start: Time for the first point in the PID data.

t_stop: Time for the last point in the PID data.

data: Data points for the PID.

time: Times corresponding to the data points for the PID.

PumpCont (defined in dbAccess.h)

```
typedef struct {
    TSTRING
    hpotp_cont[4];
} PumpCont;
```

This structure is used in retrieving the identifier of the HPOT pump contractor from the TST_HW database table, which is required to determine the pump type.

Member:

hpotp_cont: Identifier of the HPOT pump contractor.

RedundantSensor (defined in dbAccess.h)

```
typedef struct {
    TSTRING
    name[41],
    test_id[13],
    pid[13];
} RedundantSensor;
```

This structure is used in retrieving the name of the good PID for a given test corresponding to a particular PID descriptor from the RED_S_C table of the TekBase database SSME_DB.

Members:

name: Descriptor for the desired PID.

test_id: Test ID for which the PID is required.

pid: Name of the good PID corresponding to the specified descriptor.

6.5 Functions

closeDB (declaration in dbAccess.h, definition in dbAccess.c)

```
void closeDB ()
```

This function closes the TekBase query session which was started by a call to initDB. It is called once in the external effects module after all TekBase queries have been completed.

dbCloseSSME (declaration in dbAccess.h, definition in dbAccess.c)

```
void dbCloseSSME ()
```

This function closes the SSME_DB database when it is no longer needed by the program. It should be called once after all calls to the functions dbGetCompInfo, dbFindPidName, and dbFindHWDelta have been completed.

dbFindPidName (declaration in dbAccess.h, definition in dbAccess.c)

```
void dbFindPidName (char *testID, char *pidDescriptor,
                    char pidName[])
```

This function retrieves the name of a good PID for a specified test corresponding to a given PID descriptor from the RED_S_C table of the SSME_DB database. The function dbInitSSME must have been called once by the program before the first call to this function.

Arguments:

testID: Test ID for the test for which the PID is required (input).

pidDescriptor: Descriptor for the PID required (input).

pidName: Name of the good PID for the specified test corresponding to the given PID descriptor (output).

There is no return value. If a good PID is found, the name of that PID is copied into the string pidName. Otherwise, the first character of pidName is set to zero.

dbFreeGains (declaration in dbAccess.h, definition in dbAccess.c)

```
void dbFreeGains (gainsTbl *gain)
```

This function frees the memory that was allocated by the function dbGetGains.

Argument:

gain: Table of gains and least-square coefficients for each dependent PID (input).

dbGetCompInfo (declaration in dbAccess.h, definition in dbAccess.c)

```
int dbGetCompInfo (char *currentTest, char *comparisonTest)
```

This function queries the SSME_DB database to determine the comparison test and time interval to be used for the current test.

Arguments:

currentTest: Test ID for the current test (input).

comparisonTest: Test ID for the comparison test (output).

Returns a non-zero value if the function was successful, zero otherwise. The function is successful if an entry for the current test is found in the CMP_DESC table of the SSME_DB database, in which case the test ID for the comparison test is copied into the string comparisonTest. If more than one entry for the current test is found, only the one returned as the first row by the database query is used.

dbGetGains (declaration in dbAccess.h, definition in dbAccess.c)

```
void dbGetGains (char *testPump, gainsTbl **gain,  
int *numberOfEntry)
```

This function queries the EXTERNAL database to retrieve the appropriate gains and least-square coefficients, corresponding to the pump type used in the current test, required to remove the effects of the independent PIDs from each dependent PID.

Arguments:

testPump: Type of pump used in the current test (input).

gain: Table of gains and least-square coefficients for each dependent PID (output).

numberOfEntry: The number of entries in the gains table (output).

There is no return value. If the gains table is found, gain will be set to the address of an array of pointers to structures of type gainsTbl and numberOfEntry will be set to the number of entries in the array. Otherwise, gain will be set to NULL and numberOfEntry will be set to zero.

dbGetPumpType (declaration in dbAccess.h, definition in dbAccess.c)

```
void dbGetPumpType (char *currentTest, char *testPump)
```

This function finds the pump type used on the current test for determining the proper set of gains values to use in calculating the effects of the independent PIDs on the dependent PIDs.

Arguments:

currentTest: The identifier for the current test (input).

testPump: The contractor which built the HPOT pump used in the current test (output; space must be allocated by the calling function).

There is no return value. If the pump type cannot be found, a NULL string will be written to testPump.

dbInitSSME (declaration in dbAccess.h, definition in dbAccess.c)

```
void dbInitSSME ()
```

This function opens the SSME_DB database for subsequent accesses by the functions dbGetCompInfo, dbFindPidName, and dbFindHWDelta. dbInitSSME must be called by the program once before any calls are made to these three functions.

FindLowRate (declaration in external.h, definition in external.c)

```
LowRate * FindLowRate (DataBase *comp, DataBase *cur)
```

This function checks each of the independent PIDs for the current test and comparison test and determines which PID has the lowest sampling rate. The rate, number of points, and time array for that PID are stored for later use by the function Nterp.

Arguments:

comp: Comparison test information (input).

cur: Current test information (input).

Returns a structure containing the rate, number of points, and time array for the independent PID with the lowest sampling rate.

freeDBInfo (declaration in loadData.h, definition in loadData.c)

```
void freeDBInfo (DataBase *db)
```

This function frees the memory that was allocated by the function loadDBInfo and any memory allocated for independent PIDs (pointed to by elements of the member pid of the structure db) by calls to loadPid.

Argument:

db: Previously loaded information about a particular test (input).

freePid (declaration in loadData.h, definition in loadData.c)

```
void freePid (pidInfo *pid)
```

This function frees the memory that was allocated by the function loadPid.

Argument:

pid: Previously loaded information about a particular PID (input).

HandleError (declaration and definition in dbAccess.c)

```
static void HandleError ()
```

This function is provided to print a message to stderr and exit the program if an error is encountered while retrieving any information from a TekBase database.

initDB (declaration in dbAccess.h, definition in dbAccess.c)

```
void initDB ()
```

This function initializes the TekBase query session for the program. It is called once at the beginning of the external effects module before any other functions from dbAccess.c are called. The environment variable QYHOST is checked to determine which machine is the TekBase server. If the environment variable is not set, the value of the constant DB_SERVER will be used.

loadDBInfo (declaration in loadData.h, definition in loadData.c)

```
DataBase * loadPid (char *db)
```

This function retrieves the information about a specified test and the associated data files.

Argument:

db: Name of the test; this may be the test ID or the name of a particular test file (input).

Returns the available information about the test and associated data files. A structure of type DataBase is allocated and all members of that structure except pid are set. The function freeDBInfo is called when the structure is no longer needed to free the memory allocated by loadDBInfo.

loadPid (declaration in loadData.h, definition in loadData.c)

```
pidInfo * loadPid (DataBase *db_ptr, char *pid, float time[])
```

This function retrieves the data associated with a specified PID for a given test.

Arguments:

db_ptr: Database information for the test for which the PID is required (input).

pid: Name of PID required (input).

time: Time interval for which PID data is required (input).

Returns the available data from the database for the requested PID. A structure of type loadPid is allocated and all members of that structure are set. The function freePid is called when the structure is no longer needed to free the memory allocated by loadPid.

Nterp (declaration in external.h, definition in external.c)

```
void Nterp (LowRate *low, pidInfo *high)
```

This function selects the points from a PID that correspond to the times for the PID with the lowest rate as determined by the function FindLowRate.

Arguments:

low: Structure containing the rate, number of points, and times for the points for the PID with the lowest rate as determined by the function FindLowRate (input).

high: PID to be resampled at the possibly lower rate (input and output).

There is no return value, but high->points, high->data, and high->time are modified to reflect the resampling results.

pefloat (declaration and definition in writeNPID.c)

```
static float pefloat (float x)
```

This function is used by writeNPID to convert floating point values to the format required for SSME format binary data files.

Argument:

x: Floating point number to be converted (input).

Returns the converted floating point number.

removeH2PEffect (declaration in removeEffect.h, definition in removeEffect.c)

```
void removeH2PEffect (pidInfo *deltaPid, pidInfo *H2PDelta,  
                      gainsTbl *gains)
```

This function removes the effects of the independent PID LPFP inlet pressure from the provided delta PID.

Arguments:

deltaPid: Delta PID from which the LPFP inlet pressure effects are to be removed (input and output).

H2PDelta: The difference in the LPFP inlet pressure PIDs for the current and comparison tests (input).

gains: The gains values and other information for the current dependent PID (input).

There is no return value. The values in deltaPid are modified to reflect the removal of the LPFP inlet pressure effects.

removeH2TEffect (declaration in removeEffect.h, definition in removeEffect.c)

```
void removeH2TEffect (pidInfo *deltaPid, pidInfo *H2TDelta,  
                      gainsTbl *gains)
```

This function removes the effects of the independent PID LPFP inlet temperature from the provided delta PID.

Arguments:

deltaPid: Delta PID from which the LPFP inlet temperature effects are to be removed (input and output).

H2TDelta: The difference in the LPFP inlet temperature PIDs for the current and comparison tests (input).

gains: The gains values and other information for the current dependent PID (input).

There is no return value. The values in deltaPid are modified to reflect the removal of the LPFP inlet temperature effects.

removeMixRatioEffect (declaration in removeEffect.h, definition in removeEffect.c)

```
void removeMixRatioEffect (pidInfo *deltaPid,  
                           pidInfo *MixRatioDelta,  
                           gainsTbl *gains)
```

This function removes the effects of the independent PID mixture ratio from the provided delta PID.

Arguments:

deltaPid: Delta PID from which the power level effects are to be removed (input and output).

MixRatioDelta: The difference in the mixture ratio PIDs for the current and comparison tests (input).

gains: The gains values and other information for the current dependent PID (input).

There is no return value. The values in **deltaPid** are modified to reflect the removal of the mixture ratio effects.

removeO2PEffect (declaration in **removeEffect.h**, definition in **removeEffect.c**)

```
void removeO2PEffect (pidInfo *deltaPid, pidInfo *O2PDelta,
                     gainsTbl *gains)
```

This function removes the effects of the independent PID LPOP inlet pressure from the provided delta PID.

Arguments:

deltaPid: Delta PID from which the LPOP inlet pressure effects are to be removed (input and output).

O2PDelta: The difference in the LPOP inlet pressure PIDs for the current and comparison tests (input).

gains: The gains values and other information for the current dependent PID (input).

There is no return value. The values in **deltaPid** are modified to reflect the removal of the LPOP inlet pressure effects.

removeO2TEffect (declaration in **removeEffect.h**, definition in **removeEffect.c**)

```
void removeO2TEffect (pidInfo *deltaPid, pidInfo *O2TDelta,
                     gainsTbl *gains)
```

This function removes the effects of the independent PID LPOP inlet temperature from the provided delta PID.

Arguments:

deltaPid: Delta PID from which the LPOP inlet temperature effects are to be removed (input and output).

O2TDelta: The difference in the LPOP inlet temperature PIDs for the current and comparison tests (input).

gains: The gains values and other information for the current dependent PID (input).

There is no return value. The values in **deltaPid** are modified to reflect the removal of the LPOP inlet temperature effects.

removePowerLevelEffect (declaration in **removeEffect.h**, definition in **removeEffect.c**)

```
void removePowerLevelEffect (pidInfo *deltaPid,
                             pidInfo *PowerLevelCurrent,
                             pidInfo *PowerLevelComparison,
                             gainsTbl *gains)
```

This function removes the effects of the independent PID power level from the provided delta PID. The power level effects are filtered before subtracting them from the delta PID.

Arguments:

deltaPid: Delta PID from which the power level effects are to be removed (input and output).

PowerLevelCurrent: The power level PID for the current test (input).

PowerLevelComparison: The power level PID for the comparison test (input).

gains: The gains values and other information for the current dependent PID (input).

There is no return value. The values in **deltaPid** are modified to reflect the removal of the power level effects.

smoothData (declaration in **smoothData.h**, definition in **smoothData.c**)

```
int smoothData (float *data, float *smoothData, int numpts,
               int smoothWindow)
```

This function filters the array data by computing a sliding average for each point in the array, in effect a simple convolution filter.

Arguments:

data: Array of data to be filtered (input).

smoothedData: Array in which filtered data is to be stored (output).

numPts: Number of data points in the array to be filtered (input).

smoothWindow: Width of the sliding window to used when computing the averages if odd; if even, actual window size used will be **smoothWindow + 1** (input).

Returns -1 if **smoothWindow > numPts**, 0 otherwise. If the return value is 0, the array **smoothedData** will contain the filtered data, except for the first and last **smoothWindow/2** points, which will be identical to the corresponding points in the array data.

writeNPID (declaration in **writeNPID.h**, definition in **writeNPID.c**)

```
int writeNPID (char *testid, float time[2], float cutoff,
              char test_title[41], int numpids,
              pidInfo *pid_to_store, int normalize)
```

This function writes the binary SSME format data file containing the delta PIDs or normalized delta PIDs and writes some useful information to stdout.

Arguments:

testid: Identifier for the current test (input).

time: Time interval for which PID data was generated (input).

cutoff: Engine cutoff time for the current test (input).

test_title: Test title to be used for the SSME format data file (input).

numpids: Number of PIDs to be written to the data file (input).

pid_to_store: Array of structures containing the data for each of the PIDs to be written to the data file (input).

normalize: Flag to indicate whether the PIDs to be written are normalized; value must be zero for unnormalized delta PIDs and one for normalized delta PIDs (input).

Returns FALSE if an error is encountered while writing the file, TRUE otherwise.

6.6 Algorithm

The effects of each of the 6 independent PIDs on 25 dependent PIDs have been estimated and stored in the GAINS table of the EXTERNAL database for tests using the Rocketdyne HPOT pump and for tests using the ATD HPOT pump. The estimations were determined by assuming that the effects of mixture ratio, LPFP inlet pressure, LPOP inlet pressure, LPFP inlet temperature, and LPOP Inlet temperature were linear on each dependent PID and that the effect of the power level could be approximated by doing a least-squares fit to a fifth degree polynomial. The estimations of the effects are used by the External Effects Module to remove the effects of the independent PIDs from the dependent PIDs.

6.6.1 Notation

PL: Linear gain associated with the Power Level (not used because the power level effect is assumed to be nonlinear).

MR: Linear gain associated with the Mixture Ratio.

H2P: Linear gain associated with the LPFP Inlet Pressure.

O2P: Linear gain associated with the LPOP Inlet Pressure.

H2T: Linear gain associated with the LPFP Inlet Temperature.

O2T: Linear gain associated with the LPOP Inlet Temperature.

LSQCOF_1: Constant term of the fifth degree polynomial approximating the power level effects.

LSQCOF_2: Coefficient of the first degree term of the fifth order polynomial approximating the power level effects.

LSQCOF_3: Coefficient of the second degree term of the fifth order polynomial approximating the power level effects.

LSQCOF_4: Coefficient of the third degree term of the fifth order polynomial approximating the power level effects.

LSQCOF_5: Coefficient of the fourth degree term of the fifth order polynomial approximating the power level effects.

LSQCOF_6: Coefficient of the fifth degree term of the fifth order polynomial approximating the power level effects.

t: Time--a particular point in time during a test.

Dependent (t): The value of the Dependent PID at time t.

DependentComparison(t): The value of a Dependent PID at time t of the comparison test.

DependentCurrent(t): The value of a Dependent PID at time t of the current test.

PLValueCurrent(t): Value of the Independent PID Power Level of the current test at time t.

MRValueCurrent(t): Value of the Independent PID Mixture Ratio of the current test at time t.

H2PValueCurrent(t): Value of the Independent PID LPFP Inlet Pressure of the current test at time t.

O2PValueCurrent(t): Value of the Independent PID LPOP Inlet Pressure of the current test.

H2TValueCurrent(t): Value of the Independent PID LPFP Inlet Temperature of the current test.

O2TValueCurrent(t): Value of the Independent PID LPOP Inlet Temperature of the current test.

PLValueComparison(t): Value of the Independent PID Power Level of the comparison test at time t.

MRValueComparison(t): Value of the Independent PID Mixture Ratio of the comparison test at time t.

H2PValueComparison(t): Value of the Independent PID LPFP Inlet Pressure of the comparison test at time t.

O2PValueComparison(t): Value of the Independent PID LPOP Inlet Pressure of the comparison test at time t.

H2TValueComparison(t): Value of the Independent PID LPFP Inlet Temperature of the comparison test at time t.

O2TValueComparison(t): Value of the Independent PID LPOP Inlet Temperature of the comparison test at time t.

PLEffect(t): The power level effect to be subtracted from the dependent delta in calculating the normalized delta PID.

DependentNormal0(t): Intermediate value used during calculation of normalized delta PID value at time t.

DependentNormal1(t): Intermediate value used during calculation of normalized delta PID value at time t.

Normal(t): Normalized delta PID value at time t.

HWDelta: Constant to be added to the normalized delta PID to adjust for differences in the hardware configurations for the two tests.

6.6.2 Computations

Calculate all the PID deltas.

$$\begin{aligned} \text{DependentDelta}(t) &= \text{DependentCurrent}(t) - \text{DependentComparison}(t) \\ \text{PLDelta}(t) &= \text{PLValueCurrent}(t) - \text{PLValueComparison}(t) \\ \text{MRDelta}(t) &= \text{MRValueCurrent}(t) - \text{MRValueComparison}(t) \\ \text{H2PDelta}(t) &= \text{H2PValueCurrent}(t) - \text{H2PValueComparison}(t) \\ \text{O2PDelta}(t) &= \text{O2PValueCurrent}(t) - \text{O2PValueComparison}(t) \\ \text{H2TDelta}(t) &= \text{H2TValueCurrent}(t) - \text{H2TValueComparison}(t) \\ \text{O2TDelta}(t) &= \text{O2TValueCurrent}(t) - \text{O2TValueComparison}(t) \end{aligned}$$

Subtract linear Independent PID effects:

$$\begin{aligned} \text{DependentNormal0}(t) &= \text{DependentDelta}(t) - \\ &\quad (\text{MRDelta}(t) * \text{MRCoefficient} + \\ &\quad \text{H2PDelta}(t) * \text{H2PCoefficient} + \\ &\quad \text{O2PDelta}(t) * \text{O2PCoefficient} + \\ &\quad \text{H2TDelta}(t) * \text{H2TCoefficient} + \\ &\quad \text{O2TDelta}(t) * \text{O2TCoefficient}) \end{aligned}$$

Calculate Power Level effects:

```

PLEffect(t) = (LSQCOF_1 +
               LSQCOF_2 * PLValueCurrent(t) +
               LSQCOF_3 * PLValueCurrent(t)**2 +
               LSQCOF_4 * PLValueCurrent(t)**3 +
               LSQCOF_5 * PLValueCurrent(t)**4 +
               LSQCOF_6 * PLValueCurrent(t)**5) -
               (LSQCOF_1 +
               LSQCOF_2 * PLValueComparison(t) +
               LSQCOF_3 * PLValueComparison(t)**2 +
               LSQCOF_4 * PLValueComparison(t)**3 +
               LSQCOF_5 * PLValueComparison(t)**4 +
               LSQCOF_6 * PLValueComparison(t)**5)

DependentNormal1(t) = DependentNormal0(t) - PLEffect(t)

Normal(t) = DependentNormal1(t) + HWDelta

```

In addition to the calculations shown above, the program utilizes a filter on some of the intermediate calculations to reduce the effects of noise in the data. The filter is a simple convolution filter calculated as follows:

$$\text{Filtered}(t) = (\text{Data}(t - \text{WIN}/2) + \text{Data}(t + 1 - \text{WIN}/2) + \dots + \text{Data}(t + \text{WIN}/2)) / \text{WIN}$$

where WIN is the window size, which must be odd, for the filter, Data(t) is the data to be filtered, and Filtered(t) is the filtered value at time t. Currently, the following intermediate calculations are filtered: MRDelta(t), H2PDelta(t), O2PDelta(t), H2TDelta(t), O2TDelta(t), PLEffect(t) (all using a window size of SMOOTH_WIN_I), and DependentDelta(t) (using a window size of SMOOTH_WIN_D).

6.7 Cflow output

```

1  main: int(), <external.c 33>
4      initDB: void(), <dbAccess.c 68>
9          tql_init: <>
12         tql_error: <>
13         HandleError: void(), <dbAccess.c 556>
14             tql_status: <>
15             tql_mess: <>
17             tql_term: <>
19             tql_mode: <>
20             tql_create_buffer: <>
21         flInitLib: <>
22         dbInitSSME: void(), <dbAccess.c 247>
23             tql_query: <>
24             tql_error: 12
25             HandleError: 13
27         dbGetCompInfo: int(), <dbAccess.c 334>
31             tql_query: 23
32             tql_error: 12
33             HandleError: 13
34             tql_status: 14
35             tolower: <>
37         dbGetPumpType: enum(), <dbAccess.c 274>
41             tql_query: 23
42             tql_error: 12
43             HandleError: 13
44             tql_status: 14
45         dbCloseSSME: void(), <dbAccess.c 513>
46             tql_query: 23
47         dbGetGains: void(), <dbAccess.c 129>
50             tql_query: 23
51             tql_error: 12
52             HandleError: 13

```

```

53     tq1_status: 14
56     loadDBInfo: struct*(), <loadData.c 63>
58     flFindDataFile: <>
62     init_file: <>
64     flDBdirTestName: <>
66     flDBdirEntries: <>
67     flDBdirNthEntry: <>
69     dbFindPidName: void(), <dbAccess.c 398>
73     tq1_query: 23
74     tq1_error: 12
75     HandleError: 13
76     tq1_status: 14
79     loadPid: struct*(), <loadData.c 216>
83     dbread: <>
85     smoothData: int(), <smoothData.c 56>
87     release_data: <>
88     FindLowRate: struct*(), <external.c 454>
91     Nterp: void(), <external.c 499>
98     smoothData: 85
100    freePid: void(), <loadData.c 309>
102    removeH2PEffect: void(), <removeEffect.c 168>
103    removeO2PEffect: void(), <removeEffect.c 202>
104    removeH2TEffect: void(), <removeEffect.c 236>
105    removeO2TEffect: void(), <removeEffect.c 270>
106    removePowerLevelEffect: void(), <removeEffect.c 61>
108    smoothData: 85
110    removeMixRatioEffect: void(), <removeEffect.c 134>
111    dbFindHWDelta: void(), <dbAccess.c 459>
115    tq1_query: 23
116    tq1_error: 12
117    HandleError: 13
118    tq1_status: 14
119    writeNPID: int(), <writeNPID.c 66>
126    pefloat: float(), <writeNPID.c 271>
134    dbFreeGains: void(), <dbAccess.c 223>
136    freeDBInfo: void(), <loadData.c 170>
138    release_file: <>
139    freePid: 100
140    closeDB: void(), <dbAccess.c 534>
141    tq1_term: 17

```

7. HPOTP Module

This section describes the overall architecture of the Enhanced HPOTP Diagnostic System. The enhanced system is designed to run in one of two execution modes: interactive and batch. In interactive mode, the system queries a user for all needed information, computes all features dynamically as-needed, and outputs results in a textual form to the terminal. Batch mode is designed for use with the PTDS. In batch mode features are first computed by a separate module and stored in a database, then the HPOTP modules is started, reads the features in, performs its analyses and writes its results back out to the database. A user can then browse the system's results via a graphical user interface.

7.1 Modules

The major modules in the diagnostic system are each described in the following sections.

7.1.1 Executive

The executive module is primarily responsible for guiding execution of the diagnostic system through several major steps or "phases". These phases are:

initialize	Connect to database. Get test IDs and thrust profiles.
SVAL_hard_failures	Check for hard sensor failures.

SVAL_soft_failures	Determine preferred sensors via voting.
get_features	Determine features needed for diagnosis.
find_event_intervals	Determine time intervals to analyze.
find_anomalies	Diagnosis.
prepare_output	Prepare anomaly and supporting plot descriptions.
output_anomalies	Output results.
wrapup	Disconnect from database, cleanup.

7.1.2 Feature Extraction

This module obtains features requested by other parts of the HPOTP system. The requests (in the form of 'GetFeature' facts) are honored by either importing features from TekBase (in batch processing mode) or by computing them on-the-fly as needed (in interactive mode). Resulting features are stored as facts for use by redundancy management, sensor validation, and diagnostic routines.

All calls to import data from TekBase are located within this module.

7.1.3 HPOTP Sensor Validation

The HPOTP sensor validation module is responsible for detecting and diagnosing instrumentation anomalies and failures. It first attempts to import any results obtained from the PTDS sensor validation module by examining the "RED_S_C" (redundant sensor choice) table entries asserted in the SSME_DB database for the current test. If a sensor appears in the RED_S_C table, then the HPOTP module does not validate it, and takes the PTDS sensor validation module's recommendation for the best sensor to use for the specified parameter. If sensor validation information is not available for the current test under analysis (e.g., in performing an interactive mode analysis of a new test), then the HPOTP system will attempt to validate sensors on its own.

When performing its own sensor validation, the HPOTP module operates in two phases. First, it attempts to detect obvious, "hard" failures. These include:

- No data in the file.
- Exceeds gross noise limits.
- Pre- or Post-test reasonableness limit exceedance.
- Reasonableness limit exceedance during the test.
- Sensor trace is flat during the test (for sensor whose values are expected to vary significantly). This indicates that the sensor may have been disconnected.
- Redundancy voting, when three or more redundant transducers are available.

In its second phase of operation, the sensor validation module essentially replicates the voting scheme implemented in the original HPOTP diagnostic system implementation. This strategy entails selecting a "preferred" sensor from each set of redundants on the basis of the minimum number of erratic or spike features. These sensor preferences are not relied upon to the extent they were in the original implementation; wherever possible, features are redundancy voted (see the next section).

The HPOTP module always checks for certain sensor anomalies which are particular to the HPOTP. These include "redundant sensor drift" checks which make note of any cases in which redundant sensors consistently drift apart during any steady-state power segment.

7.1.4 Redundancy Management

The redundancy management module performs redundancy management for a select group of features. Whenever one of these features is defined for a sensor (either imported from TekBase or computed dynamically), it is classified by checking it against all redundant sensors. The results of the classification can be one of the following:

For transducers with only one bridge each:

- unconfirmed If the sensor has no valid redundants.
- spurious If the sensor has valid redundants, none of which register a similar feature at the same time.
- confirmed If the sensor has a redundant which registers a similar feature at the same time. Any outlier redundants (which did not see the feature) are marked as spurious.

For transducers with two bridges:

- confirmed If seen on any bridge of 2 or more transducers. (Outlier bridges are marked as spurious.)
- unconfirmed Seen on all valid bridges of a transducer and there are no other valid transducers.
- spurious If more than one transducer is valid but only seen on one. (All bridges are marked as spurious.)

Feature equivalence is based on start times only (i.e., they must be within one second of each other), and the resulting 'confirmed' feature from two or more redundant features is formed by simply selecting one of the inputs (i.e., no attempt is made to average the values).

7.1.5 Statistics Module

The statistics module is responsible for computing statistical summaries of the parameters stored in the historical database, computing parameter values for the current test, and determining if any current test parameters are "outliers".

7.1.6 Anomaly Detection & Diagnosis

The Anomaly Detection module checks for combinations of features which are indicative of known anomalies. Results are asserted as *anomaly* records, which have descriptions of appropriate supporting plots associated with them. Results are classified as either INSTRUMENTATION, OBSERVATION, or ANOMALIES, and have a priority number associated with them indicating the degree of severity.

To simplify many of the anomaly detection rules, the test time-line is partitioned into intervals within which nothing is happening (i.e. no features start or end). The anomaly detection rules which are interested in concurrent combinations of features then simply analyze each of these time intervals separately. If the same anomaly is detected in adjacent intervals, the anomaly descriptions are combined into one spanning the entire interval.

A complete list of the diagnostic rules used in the system is given in Section 7.2.

7.1.7 Green Run Specifications Check

The Green Run Specifications module checks all Green Run requirements and creates *anomaly* records whenever any violations are detected.

7.1.8 Supporting Plot Generation

The Plot Generation module takes abbreviated descriptions of supporting plots required for detected anomalies, and expands them into the 60 fields required by the PTDS to produce plots in the graphical user interface.

7.1.9 Output of Results

The Output module takes the results of the analyses, along with information about supporting plots, and either writes them to the database for later viewing (in batch mode) or prints a textual summary to the terminal (in interactive mode). This module also updates the historical database automatically (in batch mode) or if indicated by the user (in interactive mode).

All calls to export data to TekBase are located within this module.

7.2 Anomalies Currently Detected by the HPOTP Diagnostic System

This section provides a brief, but complete listing of the anomalies currently detected by the enhanced HPOTP diagnostic system.

7.2.1 General Anomalies

Rule: anomaly5.05.1

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.05

Summary: Difference 327-328 is different between current and comparison tests.

Report: "The difference (327 - 328) is different at thrust level <PL> between this test and the previous."

Rule: anomaly5.06.1

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.06

Summary: Spike seen in 327(328), not in 328(327), and no level shift in 327,328, or 327-328.

Report: "Spike seen in sensor <327|328> only, with no change in steady state pressures or pressure difference. Possible sensor or omni seal anomaly. No real rotor motion."

Rule: anomaly5.06.2

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.06

Summary: Level shift seen in 327(328) and not in 328(327).

Report: "Level shift seen in <327|328> only. Possible sensor problem, omni seal leakage problem. No real rotor motion."

Rule: anomaly5.06.3

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.06

Summary: Spike seen in 327 and 328, and level shift seen in 327-328.

Report: "Possible HPOTP momentary anomalous rotor motion.Possible HPOTP balance piston momentary shift in orifice position."

Rule: anomaly5.06.4

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.06

Summary: Level shift seen in 327 and 328 (opposite directions), and in 327-328.

Report: "Possible HPOTP anomalous rotor motion."

Rule: anomaly5.06.5

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.06

Summary: Level shift seen in 327 and 328 (same direction).

Report: "Possible HPOTP balance piston orifice position change."

Rule: anomaly5.06.7

Source: SAIC final report, section 5.06

Pumps: Rocketdyne and Pratt&Whitney

Summary: Level shift seen in 327(328) and not in 328(327).

Report: "Statistically significant change in <327|328> but not in difference (327 - 328). Possible omni seal leakage. No real rotor motion."

Rule: anomaly5.06.9

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.06

Summary: Level shift seen in 327-328, but not in 327 or 328.

Report: "Statistically significant change in difference (327 - 328) but not in individual sensors. Not anomalous; no real rotor motion."

Rule: anomalyRotorDrag1, anomalyRotorDrag1

Pumps: Rocketdyne and Pratt&Whitney

Source: Wilmer

Summary:

1. Concurrent: Significant increase in 328, decrease in 327, LOX in P is flat
Following increase in PL or decrease in LOX in P
Duration of more than 10 seconds (or duration of current power level, if less).
2. Opposite of above case.

Report: "Possible rotor drag."

Rule: anomaly5.07.1

Pumps: Rocketdyne only

Source: SAIC final report, section 5.07

Summary: PBP bistability detection. Just reports result from C routine.

Report: "PBP bistability at thrust level <PL>"

Rule: anomaly5.08.1

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.08

Summary: Erratic 990, 1190 not erratic or spiking.

Report: "HPOTP erratic primary turbine seal drain pressure may indicate sensor problem or seal anomaly. No effect seen in drain temperature."

Rules: ANoseLeak_one, ANoseLeak_both
Pumps: Rocketdyne only
Summary: IMSL Nose Seal Leak detected on either 211 or 212.
Source: Wilmer
Report: "Nose seal leakage indicated by PID <PID>."

Rule: anomaly5.08.2
Pumps: Rocketdyne and Pratt&Whitney
Source: SAIC final report, section 5.08
Summary: 1190 erratic, 990 not erratic or spiking.
Report: "HPOTP erratic primary turbine seal drain temperature may indicate sensor problem or seal anomaly. No effect seen in drain pressure."

Rule: anomaly5.08.3
Source: SAIC final report, section 5.08
Pumps: Rocketdyne and Pratt&Whitney
Summary: 990 erratic or spiking, and 1190 erratic or spiking.
Report: "HPOTP shows concurrent jitter in both primary turbine seal drain pressure and temperature. Possible seal anomaly."

Rule: anomaly5.08.4
Pumps: Rocketdyne and Pratt&Whitney
Source: Wilmer
Summary: Erratic or spiking 990 & 1190 in same power-level interval, but not concurrently.
Report: "HPOTP shows non-concurrent jitter in both primary turbine seal drain pressure and temperature. Possible seal anomaly."

Rule: anomaly990shift
Pumps: Rocketdyne only
Source: Wilmer
Summary: 990 is low (or high) in peak and equilibrium values relative to family.
Primary turbine seal drain pressure.
Report: "HPOTP primary turbine seal drain pressure is <HIGH|LOW> in peak and equilibrium values compared to historical statistics. May be change in seal clearance or sensor calibration."

Rule: anomalyA990peakshift
Pumps: Rocketdyne only
Source: Wilmer
Summary: 990 peak value is out-of-family, but equilibrium value is OK.
Report: "HPOTP primary turbine seal drain pressure peak is <HIGH|LOW> compared to historical statistics. May be change in seal clearance or sensor calibration."

Rule: anomaly990peakshift
Pumps: Rocketdyne only
Source: Wilmer
Summary: 990 equilibrium value is out-of-family, but peak value is OK.
Report: "HPOTP primary turbine seal drain pressure equilibrium value is <HIGH|LOW> compared to historical statistics. May be change in seal clearance or sensor calibration."

Rule: anomaly91shift
Pumps: Rocketdyne only

Source: Wilmer

Summary: 91 or 92 is low (or high) in peak and equilibrium values relative to family.

Secondary turbine seal cavity pressure.

Report: "HPOTP secondary turbine seal cavity pressure is <HIGH|LOW> in peak and equilibrium values compared to historical statistics. May be change in seal clearance or sensor calibration."

Rule: anomaly91peakshift

Pumps: Rocketdyne only

Source: Wilmer

Summary: 91/92 peak value is out-of-family, but equilibrium value is OK.

Report: "HPOTP secondary turbine seal cavity pressure peak is <HIGH|LOW> compared to historical statistics. May be change in seal clearance or sensor calibration."

Rule: anomaly91eqshift

Pumps: Rocketdyne and Pratt&Whitney

Source: Wilmer

Summary: 91/92 equilibrium value is out-of-family, but peak value is OK.

Report: "HPOTP secondary turbine seal cavity pressure equilibrium value is <HIGH|LOW> compared to historical statistics. May be change in seal clearance or sensor calibration."

Rule: anomaly91peakwidth

Pumps: Rocketdyne only

Source: Wilmer

Summary: 91/92 peak width is out-of-family.

Report: "HPOTP secondary turbine seal cavity pressure peak width is <HIGH|LOW> compared to historical statistics."

Rule: anomaly990peakwidth

Pumps: Rocketdyne only

Source: Wilmer

Summary: 990 peak width is out-of-family.

Report: "HPOTP primary turbine seal drain pressure peak width is <HIGH|LOW> compared to historical statistics."

Rule: anomaly91peaktime

Pumps: Rocketdyne only

Source: Wilmer

Summary: 91/92 peak time is out-of-family.

Report: "HPOTP secondary seal cavity pressure peak time is <HIGH|LOW> compared to historical statistics."

Rule: anomaly990peaktime

Pumps: Rocketdyne only

Source: Wilmer

Summary: 990 peak time is out-of-family.

Report: "HPOTP primary turbine seal drain pressure peak time is <HIGH|LOW> compared to historical statistics."

Rule: anomalyIMSLstart

Pumps: Rocketdyne and Pratt&Whitney

Source: Wilmer

Summary: START value of 211/212 is out-of-family.

Report: "HPOTP intermediate seal purge pressure is <HIGH|LOW> at START compared to historical statistics."

Rule: anomalyBalPistonFamily1

Pumps: Rocketdyne and Pratt&Whitney

Source: Wilmer

Summary: 327 is out-of-family (at 109MAX, 104MIN, or 104Nominal NPSP).

Report: "HPOTP balance cavity pressure A is <HIGH|LOW> at <time> compared to historical statistics. B channel is within limits."

Rule: anomalyBalPistonFamily2

Pumps: Rocketdyne and Pratt&Whitney

Source: Wilmer

Summary: 328 is out-of-family (at 109MAX, 104MIN, or 104Nominal NPSP).

Report: "HPOTP balance cavity pressure B is <HIGH|LOW> at <time> compared to historical statistics. A channel is within limits."

Rule: anomalyBalPistonFamily3

Pumps: Rocketdyne and Pratt&Whitney

Source: Wilmer

Summary: 327 and 328 are both out-of-family (at 104MIN, 109Max, or 104Nominal NPSP).

Report: "HPOTP balance cavity pressure A is <HIGH|LOW> and channel B is <HIGH|LOW> at <time> compared to historical statistics."

Rule: anomalyLOXSIPFamily

Pumps: Rocketdyne and Pratt&Whitney

Source: Wilmer

Summary: 951/952/953 are out-of-family.

Report: "HPOTP primary pump seal drain pressure is <HIGH|LOW> from 5 seconds to cutoff compared to historical statistics."

Rule: anomalyLOXSITFamily

Pumps: Rocketdyne and Pratt&Whitney

Source: Wilmer

Summary: 1187 is out-of-family.

Report: "HPOTP primary pump seal drain temperature maximum is <HIGH|LOW> compared to historical statistics."

Rule: anomalySlingerProblem

Pumps: Rocketdyne and Pratt&Whitney

Source: Wilmer

Summary: 1187 is out-of-family low and 951/952/953 is out-of-family high.

Report: "HPOTP primary pump seal drain temperature maximum is LOW and HPOTP primary pump seal drain pressure is HIGH compared to historical statistics. Indicates possible slinger problem."

Rule: anomaly5.09.6

Pumps: Rocketdyne only

Source: SAIC final report, section 5.09

Summary: Could not compute a peak for 990.

Report: "Current test HPOTP primary turbine seal drain pressure peak missing."

Rule: anomaly5.09.12

Pumps: Rocketdyne only

Source: SAIC final report, section 5.09

Summary: Could not compute a peak for 91/92.

Report: "Current test HPOTP secondary turbine seal cavity pressure peak missing."

Rule: anomaly5.12.1

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.12

Summary: 91/92 is erratic, 1188 is normal.

Report: "HPOTP erratic secondary turbine seal drain pressure may indicate sensor problem or seal anomaly. No effect seen in drain temperature."

Rule: anomaly5.12.2

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.12

Summary: 1188 is erratic, 91/92 is normal.

Report: "HPOTP erratic secondary turbine seal drain temperature may indicate sensor problem or seal anomaly. No effect seen in drain pressure."

Rule: anomaly5.12.3

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.12

Summary: 91/92 and 1188 are both erratic or spiking. Check for concurrent anomalies.

Report: "HPOTP shows concurrent jitter in both secondary turbine seal drain pressure and temperature. Possible seal anomaly."

Rule: anomaly5.12.4

Pumps: Rocketdyne and Pratt&Whitney

Source: Wilmer

Summary: 91/92 and 1188 are both erratic or spiking. Check for non-concurrent anomalies.

Report: "HPOTP shows non-concurrent jitter in both secondary turbine seal drain pressure and temperature. Possible seal anomaly."

Rule: anomaly5.15.1

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.15

Summary: 951/952/953 is erratic, 1187 is normal.

Report: "HPOTP erratic primary pump seal drain pressure may indicate sensor problem or seal anomaly. No effect seen in drain temperature."

Rule: anomaly5.15.2

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.15

Summary: 1187 is erratic, 951/952/953 are normal.

Report: "HPOTP erratic primary pump seal drain temperature may indicate sensor problem or seal anomaly. No effect seen in drain pressure."

Rule: anomaly5.15.3

Pumps: Rocketdyne and Pratt&Whitney

Source: SAIC final report, section 5.15

Summary: 951/952/953 and 1187 are both erratic or spiking. Concurrent anomaly in both sensors.
Report: "HPOTP shows concurrent jitter in both primary pump seal drain pressure and temperature.
Possible seal anomaly."

Rule: anomaly5.15.4

Pumps: Rocketdyne and Pratt&Whitney

Source: Wilmer

Summary: 951/952/953 and 1187 are both erratic or spiking. Non-concurrent anomaly in both sensors.

Report: "HPOTP shows non-concurrent jitter in both primary pump seal drain pressure and temperature.
Possible seal anomaly."

Rule: anomaly5.19.3

Pumps: Rocketdyne and Pratt&Whitney

Source: Inferred from Priority table in SAIC's POST_defs.h file.

Summary: 233/234 are erratic or spiking.

Report: "Spike or erratic behavior in HPOT discharge temperature (confirmed by two sensors)."

Rule: ADeltaPfamily

Pumps: Rocketdyne and Pratt&Whitney

Source: Inferred from examples from Wilmer.

Summary: 327-328 statistics (for any time interval) are greater than 2.5 sigma.

Report: "HPOTP balance cavity pressure delta-P is out-of-family <HIGHLOW> during <time>
conditions, compared to historical statistics."

7.2.2 Green Run Specifications

Rule: GREEN_check_duration

Source: Green Run Specs, RL00461, 6 Jan 1988

Summary: The following rules determine the total time spent at 104% and 109%, and check them against the Green Run specs.

Report: "Failed HPOTP Green Run test duration criteria 3.5.1.2(a)."

Rule: GREEN_check_LPOTP_inlet

Source: Green Run Specs, RL00461, 6 Jan 1988

Summary: The following rules check the minimum required duration at MIN and MAX LOX pressurization.

Report:

1. "Failed HPOTP Green Run LPOTP inlet criteria 3.5.1.2(b)."
"(Minimum NPSP of 20+5/-0 for 5 seconds at 104% or higher.)"
2. "Failed HPOTP Green Run LPOTP inlet criteria 3.5.1.2(c)."
"(Maximum NPSP of 150+10/-0 for 10 seconds at 104% or higher.)"

Rule: GREEN_check_65_time

Source: Green Run Specs, RL00461, 6 Jan 1988

Summary: Checks the minimum bucket durations.

Report: "Failed HPOTP Green Run 65/64/63% throttle criteria 3.5.1.2(d)"

Rule: GREEN_check_limits

Source: Green Run Specs, RL00461, 6 Jan 1988

Summary: 3513II & III, Checks the various hard Green Run limits

Report:

1. "Failed HPOTP Green Run limits at START for <parameter>."
2. "Failed HPOTP Green Run <PL>% peak limits for <parameter>."
3. "Failed HPOTP Green Run <VentCondition> limits for <parameter>."

Rule: GREEN_check_IMSLStart

Source: Wilmer

Summary: Normalized form. Checks the intermediate seal purge pressure requirement at start.

Report: "Failed HPOTP Green Run intermediate seal purge pressure START criteria."

Rule: GREEN_check_DeltaT

Source: Green Run Specs, RL00461, 6 Jan 1988, Wilmer

Summary: 3513II & III, Checks the minimum turbine delta-T requirements. If limit is exceeded and turbine temps are cold, then that is offered as an explanation.

Report:

1. "Failed HPOTP Green Run <PL>% limits for turbine Delta-T. Probable cause is cold turbine temperature (below 1300.0)."
2. "Failed HPOTP Green Run <PL>% limits for turbine Delta-T. Turbine temperature is not cold."

Rule: GREEN_check_DeltaSpeed

Source: Green Run Specs, RL00461, 6 Jan 1988

Summary: 3513II & III, Checks delta-speed requirements.

Report: "Failed HPOTP Green Run <PL>% limits for speed change."

8. Common Function Library for PTDS

This section includes descriptions of any code written for the Post-Test Diagnostic System which is used in several modules.

8.1 Source Files

DBCT_utils_t.c

RSRC_dbutils_t.c

RSRC_rlist.c

STRNG_utils.c

tbl.c: Functions which provide generalized access to non-relational, tabular data.

tekbase.c: TekBase mode functions for the generalized table interface functions in tbl.c.

tektables.c: Initialization function for TekBase mode and table declarations for the table interface functions in tbl.c.

8.2 Header Files

DBCT_defs.h:

DBFL_defs.h

RSRC_defs.h

STRNG_defs.h

tbl.h: Header file containing constant definitions, type definitions, and declarations of external functions defined in tbl.c.

tekbase.h: Header file containing declarations of external functions defined in tekbase.c.
tektables.h: Header file containing the declaration of the function defined in tektables.c.

8.3 Defined Constants

ALLCOLUMNS: value -1; defined in tekbase.c.
Boolean: Used as the type for integer variables that only take on the values True and False; value int; defined in STRNG_defs.h.
DBCT_MaxStringLength: value 200; defined in DBCT_defs.h.
DBFL_CommandTimeStringLength: value 8; defined in DBFL_defs.h.
DBFL_DateStringLength: value 6; defined in DBFL_defs.h.
DBFL_DefaultsCols: value 50; defined in DBFL_defs.h.
DBFL_EngineNumberStringLength: value 4; defined in DBFL_defs.h.
DBFL_LegendLabelStringLength: value 25; defined in DBFL_defs.h.
DBFL_LRUStringLength: value 8; defined in DBFL_defs.h.
DBFL_MaxAllowableTableNameLen: value 20; defined in DBFL_defs.h.
DBFL_MaxCommandLength: value 600; defined in DBFL_defs.h.
DBFL_MaxDBNameLength: value 20; defined in DBFL_defs.h.
DBFL_MaxDescriptionStringLength: value 38; defined in DBFL_defs.h.
DBFL_MaxExpertModuleNameLength: value 9; defined in DBFL_defs.h.
DBFL_MaxLimitTypeStr: value 6; defined in DBFL_defs.h.
DBFL_MaxMeasurementStringLength: value 60; defined in DBFL_defs.h.
DBFL_MaxModuleNameLength: value 25; defined in DBFL_defs.h.
DBFL_MaxStringLength: value 200; defined in DBFL_defs.h.
DBFL_MaxTableStringLength: value 20; defined in DBFL_defs.h.
DBFL_MaxTestIdLength: value 9; defined in DBFL_defs.h.
DBFL_MaxTimeLimitStr: value 10; defined in DBFL_defs.h.
DBFL_MaxUnitsStringLength: value 10; defined in DBFL_defs.h.
DBFL_MaxUnknownParamLength: value 60; defined in DBFL_defs.h.
DBFL_ModuleStringLength: value 10; defined in DBFL_defs.h.
DBFL_ObservationStringLength: value 200; defined in DBFL_defs.h.
DBFL_PidNameLength: value 12; defined in DBFL_defs.h.
DBFL_PIDStringLength: value 12; defined in DBFL_defs.h.
DBFL_PlotSubTitleStringLength: value 40; defined in DBFL_defs.h.
DBFL_PlotTitleStringLength: value 40; defined in DBFL_defs.h.

DBFL_PlotXTitleStringLength: value 40; defined in DBFL_defs.h.
 DBFL_PlotYTitleStringLength: value 40; defined in DBFL_defs.h.
 DBFL_PostulateStringLength: value 500; defined in DBFL_defs.h.
 DBFL_ResourceNameStringLength: value 20; defined in DBFL_defs.h.
 DBFL_TestIdStringLength: value 6; defined in DBFL_defs.h.
 DBFL_TimeStringLength: value 25; defined in DBFL_defs.h.
 False: value 0; defined in STRNG_defs.h.
 FALSE: value 0; defined in tbl.h, tbl.c, and tekbase.c.
 HASHSIZE: value 101; defined in tekbase.c.
 MAX_COMMANDS: value 100; defined in tbl.h.
 MAXTNAME: value 20; defined in tekbase.c.
 RSRC_ExecEnvVar: value "NASA_HOME"; defined in RSRC_defs.h.
 RSRC_Incomplete: value 2; defined in RSRC_defs.h.
 RSRC_MaxNumResources: value 150; defined in RSRC_defs.h.
 RSRC_NumResourcesPerInt: value 15; defined in RSRC_defs.h.
 RSRC_Off: value 0; defined in RSRC_defs.h.
 RSRC_On: value 1; defined in RSRC_defs.h.
 RSRC_ResourceBoardUpdateStringLength: value 200; defined in RSRC_defs.h.
 RSRC_ResourceNumBits: value 2; defined in RSRC_defs.h.
 TBL_END: value NULL; defined in tbl.h.
 TBL_NUM_MODES: The number of database modes available, corresponding to the number of values for the enum type tbl_modes (see section 4.4); value 3; defined in tbl.h.
 True: value 1; defined in STRNG_defs.h.
 TRUE: value 1; defined in tbl.h, tbl.c, and tekbase.c.
 TUSER: value "GUEST"; defined in tekbase.c.

8.4 Defined Types

accessfun (defined in tbl.c)

```
typedef int (*accessfun)();
```

This type is used for convenience in declaring pointers to the mode functions for database access.

mode_info (defined in tbl.c)

```
struct mode_info {
    accessfun
    countfun,
    putfun,
```

```

        getfun,
        deletefun,
        updatefun,
        freefun,
        donefun;
    int
        in_use;
};

```

This structure is used to store pointers to the mode functions for database access.

Members:

countfun: Pointer to the mode function which counts the number of rows in a table which meet specified conditions.

putfun: Pointer to the mode function which adds a row of data to a table.

getfun: Pointer to the mode function which retrieves data from a table.

deletefun: Pointer to the mode function which deletes one or more rows of data from a table.

updatefun: Pointer to the mode function which changes values in a table.

freefun: Pointer to the mode function which frees any memory which was allocated by the other table access functions; this function is called once for each referenced table after access to the database is no longer needed.

donefun: Pointer to the mode function which performs any additional functions required when access to the database is no longer needed; this function is called once after all calls to freefun are complete.

in_use: Flag used to indicate that the mode corresponding to the functions is being used by the program.

RSRC_ResourceBoard and RSRC_PResourceBoard (defined in RSRC_defs.h)

```

typedef struct RSRC_resourceboard {
    char
        test_id[DBFL_TestIdStringLength+1];
    int
        resource[RSRC_MaxNumResources];
} RSRC_ResourceBoard, *RSRC_PResourceBoard;

```

RSRC_ResourceList and RSRC_PResourceList (defined in RSRC_defs.h)

```

typedef struct RSRC_resourcelist {
    int
        id;
    char
        name[DBFL_ResourceNameStringLength+1];
} RSRC_ResourceList, *RSRC_PResourceList;

```

sort_ops (defined in tbl.h)

```

enum sort_ops {TASC=1, TDESC};

```

tbl_column (defined in tbl.h)

```

struct tbl_column {

```

```

    enum tbl_data_types
        data_type;
    char
        external_name[80],
        internal_name[80];
}

```

tbl_command (defined in tbl.h)

```

struct tbl_command {
    int
        column;
    enum tbl_ops
        command;
    void
        *arg;
}

```

tbl_data_types (defined in tbl.h)

```

enum tbl_data_types {TBL_STRING=1, TBL_INT, TBL_FLOAT};

```

tbl_info (defined in tbl.h)

```

struct tbl_info {
    char
        internal_name[80],
        external_name[80];
    enum tbl_modes
        mode;
    char
        DB_name[80];
    int
        num_columns;
    struct tbl_column
        *columns;
    struct tbl_info
        *next;
    void
        *data;
}

```

tbl_modes (defined in tbl.h)

```

enum tbl_modes {TBL_TEXT=1, TBL_TEKBASE, TBL_INGRES};

```

These are the database modes available to the program. PTDS currently uses TekBase, although code exists for other modes used previously. If other modes are added in the future, this type and the constant TBL_NUM_MODES must be updated.

Values:

TBL_TEXT: Mode for databased information stored in flat ASCII files.

TBL_TEKBASE: Mode for use of TekBase databases.

TBL_INGRES: Mode for use of INGRES databases.

tbl_ops (defined in tbl.h)

```
enum tbl_ops {TEQ=1, TGE, TLT, TGT, TLE, TNE, TFETCH, TUPDATE};
```

tlist (defined in tekbase.c)

```
struct tlist {
    char
        typename[MAXTNAME];
    TSHORT
        valtype;
    TLONG
        fieldsize;
}
```

TQL_type (defined in tekbase.c)

```
struct TQL_type {
    TSHORT
        valtype;
    TLONG
        fieldsize;
}
```

unique_values (defined in tekbase.c)

```
enum unique_values {UNQ_NONE, UNQ_TRUE, UNQ_FALSE};
```

8.5 Global Variables

command_string: static char [1000]; defined in tekbase.c.

condition_string: static char [1000]; defined in tekbase.c.

connected: static int; initial value FALSE; defined in tekbase.c.

current_DB: static char [80]; defined in tekbase.c.

fetch_commands: static int *; defined in tekbase.c.

fetch_eof: static int *; defined in tekbase.c.

fetch_count: static int *; defined in tekbase.c.

fetch_numcommands: static int; defined in tekbase.c.

fetch_numconditions: static int; defined in tekbase.c.

fetch_tbl: static struct tbl_info *; defined in tekbase.c.

mode_list: Pointers to the mode functions for each database mode in use; static struct mode_info [TBL_NUM_MODES]; defined in tbl.c.

num_select_params: static int; initial value 0; defined in tekbase.c.

qbuffptr: Buffer used for database access by all functions using TekBase databases; char *; defined in tbl.h.

RSRC_NumDisplayableResources: int; defined in RSRC_defs.h.
 RSRC_NumResources: int; defined in RSRC_defs.h.
 RSRC_WhatIfResources: int; defined in RSRC_defs.h.
 RSRC_TheDisplayableResourceList: RSRC_ResourceList *; defined in RSRC_defs.h.
 RSRC_TheResourceList: RSRC_ResourceList *; defined in RSRC_defs.h.
 RSRC_TheWhatIfResourceList: RSRC_ResourceList *; defined in RSRC_defs.h.
 tbl_commands: struct tbl_command [MAX_COMMANDS]; declared in tbl.h and defined in tbl.c.
 tbl_conditions: struct tbl_command [MAX_COMMANDS]; declared in tbl.h and defined in tbl.c.
 tbl_list: A list of tables accessed by the program and associated data; struct tbl_info *; initial value NULL; declared in tbl.h and defined in tbl.c.
 tbl_numcommands: int; declared in tbl.h and defined in tbl.c.
 tbl_numconditions: int; declared in tbl.h and defined in tbl.c.
 tpvalues: static char[MAX_COMMANDS][100]; defined in tekbase.c.
 TQL_string: static char [1000]; defined in tekbase.c.
 TQL_types: struct TQL_type [100]; defined in tekbase.c.
 TQL_typecache: struct tlist [HASHSIZE]; defined in tekbase.c.
 unique_state: static enum unique_values; initial value UNQ_NONE; defined in tekbase.c.
 write_ptr: static char *; defined in tekbase.c.

8.6 Functions

cache_type (declaration and definition in tekbase.c)

```
static void cache_type (char *name, TSHORT valtype,
                       TLONG fieldsize)
```

This function .

Arguments:

```
name:
valtype:
fieldsize:
```

check_DB (declaration and definition in tekbase.c)

```
static int check_DB (struct tbl_info *tbl)
```

This function ensures that the process is connected to TekBase and the database for the specified table is open.

Argument:

tbl: The current database table (input).

Returns TRUE if OK or FALSE if any problems are encountered.

clear_typecache (declaration and definition in tekbase.c)

```
static void clear_typecache ()
```

This function .

clear_unique (declaration and definition in tekbase.c)

```
static void clear_unique ()
```

This function .

DBCT_DBConnect (declaration in DBCT_defs.h, definition in DBCT_utils_t.c)

```
void DBCT_DBConnect (char *database)
```

This function opens a specified database.

Argument:

database: The database to be opened (input).

DBCT_DBDisconnect (declaration in DBCT_defs.h, definition in DBCT_utils_t.c)

```
void DBCT_DBDisconnect ()
```

This function closes the current database session.

DBCT_DBSessionConnect (declaration in DBCT_defs.h, definition in DBCT_utils_t.c)

```
void DBCT_DBSessionConnect (char *database, int session)
```

This function opens the specified database.

Arguments:

database: The database to be opened (input).

session: The session ID for the database session; currently not used as it is not applicable to Metrica 3.1 (input).

DBCT_DBSessionDisconnect (declaration in DBCT_defs.h, definition in DBCT_utils_t.c)

```
void DBCT_DBSessionDisconnect (int session)
```

This function disconnects from the database with the specified session ID.

Arguments:

session: The session ID for the database session to be closed; currently not used as it is not applicable to Metrica 3.1 (input).

DBCT_SetDBSession (declaration in DBCT_defs.h, definition in DBCT_utils_t.c)

```
void DBCT_SetDBSession (int session)
```

This function is a holdover from the Ingres version of the database utilities. It is not applicable to Metrica 3.1 and is not used by the Session Manager.

DBIO_RemoveTrailingSpaces (declaration in DBCT_defs.h, definition in DBCT_utils_t.c)

```
char *DBIO_RemoveTrailingSpaces (char *string)
```

This function removes trailing white spaces from a character array.

Argument:

string: The character string from which trailing white spaces are to be removed (input and output).

The return value is the argument string. The character string is modified by overwriting all trailing white space with NULL characters.

find_col (declaration in tbl.h, definition in tbl.c)

```
int find_col (struct tbl_info *tbl, char *name)
```

This function .

Arguments:

tbl: Pointer to a table (input).

name: The internal name of a column (input).

The return value is the column number associated with name or -1 if the column is not found.

find_tbl (declaration in tbl.h, definition in tbl.c)

```
struct tbl_info *find_tbl (char *name)
```

This function locates the information about a table with a specified internal name.

Argument:

name: The internal name of a table (input).

The return value is a pointer to the table associated with name or NULL if the table is not found.

get_cached_type (declaration and definition in tekbase.c)

```
static int get_cached_type (char *name, TSHORT *valtype,  
                           TLONG *fieldsize)
```

This function .

Arguments:

name:

valtype:

fieldsize:

get_column_command (declaration and definition in tekbase.c)

```
static struct tbl_command  
*get_column_command (struct tbl_command *commands,  
                    int numcommands, int column)
```

This function .

Arguments:

command: The current command list (input).

numcommands: The number of commands in the list (input).

column: A user specified, 1-based, column number (input).

Returns the column structure associated with the number or NULL if no column structure found.

GetConditionRelop (declaration and definition in `tekbase.c`)

```
static char *GetConditionRelop (enum tbl_ops op)
```

This function .

Argument:

op: The current TBL relational operator (input).

Returns the TekBase string representation of the operator.

handle_count (declaration and definition in `tekbase.c`)

```
static void handle_count (int rows)
```

This function is called by TekBase after it has written a long value into the query buffer in response to a TQL 'COUNT' command.

Argument:

rows: The number of rows of data currently in the data buffer (input).

handle_error (declaration and definition in `tekbase.c`)

```
static int handle_error ()
```

This function prints out the current TekBase error and disconnects from the database. It returns FALSE.

handle_fetch (declaration and definition in `tekbase.c`)

```
static void handle_fetch (int rows)
```

This function is called by TekBase with each row of data after a DISPLAY command has been issued, to retrieve and store the data values in the QueryBuffer.

Argument:

rows: The number of rows of data currently in the QueryBuffer (input).

hash (declaration and definition in `tekbase.c`)

```
static unsigned hash (char *s)
```

This function .

Argument:

s:

init_PTDS_tekbase (declaration in `tektables.h`, definition in `tektables.c`)

```
void init_PTDS_tekbase ()
```

This function initializes TekBase mode and table declarations for the table interface routines in `tbl.c`.

MakeConditionString (declaration and definition in tekbase.c)

```
static int MakeConditionString (struct tbl_info *tbl,  
                               struct tbl_command *conditions,  
                               int numconditions)
```

This function formats a legal TekBase 'WHERE' clause into the global condition_string.

Arguments:

tbl: The current table (input).
conditions: The current list of conditions (input).
numconditions: The length of the list (input).

Returns TRUE if OK or FALSE if an error is detected.

MakeParameterList (declaration and definition in tekbase.c)

```
static int MakeParameterList (struct tbl_info *tbl,  
                              struct tbl_command *conditions,  
                              int numconditions)
```

This function formats a comma-separated list of TekBase column names into the global condition_string.

Arguments:

tbl: The current table (input).
conditions: The current list of conditions (input).
numconditions: The length of the list (input).

Returns TRUE if OK or FALSE if an error is detected.

MakeStringValue (declaration and definition in tekbase.c)

```
static char *MakeStringValue (struct tbl_info *tbl, int column,  
                              void *arg)
```

This function .

Arguments:

tbl: The current table (input).
column: The table column index of the value to be converted (input).
arg: A pointer to the current value to convert (input).

Returns a string representation of the value.

MakeUpdateList (declaration and definition in tekbase.c)

```
static int MakeUpdateList (struct tbl_info *tbl,  
                           struct tbl_command *conditions,  
                           int numconditions)
```

This function formats a TekBase 'UPDATE' list of the form <value> AS <columnname> into the global condition_string.

Arguments:

tbl: The current table (input).
conditions: The current list of conditions (input).

numconditions: The length of the list (input).

Returns TRUE if OK or FALSE if an error is detected.

parse_tbl_commands (declaration and definition in `tbl.c`)

```
static int parse_tbl_commands (struct tbl_info *tbl, va_list ap)
```

This function parses the commands and conditions in the variable-length argument list into the global arrays `tbl_commands`, `tbl_conditions`, `tbl_numcommands`, and `tbl_numconditions`.

Arguments:

tbl: A pointer to the current table (input).

ap: A variable-length argument list pointer (input).

Returns TRUE if all went well or FALSE if any problems are encountered.

put_row (declaration and definition in `tekbase.c`)

```
static int put_row (struct tbl_info *tbl,  
                   struct tbl_command *commands, int numcommands)
```

This function writes data from the command list into the TekBase shared memory buffer in preparation for transfer to the TekBase database.

Arguments:

tbl: The current table (input).

commands: The current list of TBL commands (input).

numcommands: Number of commands in the list (input).

Returns TRUE if OK or FALSE if any errors are detected.

RSRC_FindResourceId (declaration in `RSRC_defs.h`, definition in `RSRC_rlist.c`)

```
int RSRC_FindResourceId (char *name)
```

This function finds the index number of the resource having a specified name.

Argument:

name: Name of the desired resource (input).

Returns the index number if found, zero otherwise.

RSRC_FindResourceName (declaration in `RSRC_defs.h`, definition in `RSRC_rlist.c`)

```
void RSRC_FindResourceName (int id, char *name)
```

This function finds the name of the resource having a specified index number.

Arguments:

id: Index number of the desired resource (input).

name: Resource name corresponding to the specified index number (output).

Returns one if the resource name is found, zero otherwise. If the resource name is found, it is copied into the array `name`, which must have been previously allocated.

RSRC_FreeDisplayableResourceList (declaration in `RSRC_defs.h`, definition in `RSRC_rlist.c`)

```
void RSRC_FreeDisplayableResourceList ()
```

This function frees the global displayable resource list.

RSRC_FreeResourceList (declaration in `RSRC_defs.h`, definition in `RSRC_rlist.c`)

```
void RSRC_FreeResourceList ()
```

This function frees the global resource list.

RSRC_GetBits (declaration in `RSRC_defs.h`, definition in `RSRC_rlist.c`)

```
int RSRC_GetBits (unsigned x, unsigned pos, unsigned num_bits)
```

This function determines the value obtained by taking a specified number of bits from a specified position in a number.

Arguments:

x: Number from which the bits are to be taken (input).

pos: Position of the low order bit to be taken (input).

num_bits: Number of bits to be taken from the number (input).

Returns the value of the required bits.

RSRC_GetDisplayableResourceListOrderedById (declaration in `RSRC_defs.h`, definition in `RSRC_dbutils.c`)

```
void RSRC_GetDisplayableResourceListOrderedById ()
```

This function .

RSRC_GetNumPreqs (default declaration, definition in `RSRC_dbutils.c`)

```
int RSRC_GetNumPreqs ()
```

This function .

RSRC_GetResourceList (declaration in `RSRC_defs.h`, definition in `RSRC_dbutils.c`)

```
void RSRC_GetResourceList ()
```

This function .

RSRC_GetResources (declaration in `RSRC_defs.h`, definition in `RSRC_dbutils.c`)

```
void RSRC_GetResources (RSRC_ResourceBoard **list_ptr,  
                        int *num_tests)
```

This function .

Arguments:

list_ptr: (output).

num_tests: (output).

RSRC_GetWhatifModules (declaration in `RSRC_defs.h`, definition in `RSRC_dbutils.c`)

```
void RSRC_GetWhatifModules ()
```

This function .

RSRC_InitializeDisplayableResourceList (declaration in `RSRC_defs.h`, definition in `RSRC_dbutils.c`)

```
void RSRC_InitializeDisplayableResourceList ()
```

This function .

RSRC_InsertResourceBoardTestId (declaration in `RSRC_defs.h`, definition in `RSRC_dbutils.c`)

```
void RSRC_InsertResourceBoardTestId (char *test_id)
```

This function adds an entry for a specified test to the `resource_board` table of the Session Manager database.

Argument:

`test_id`: Test ID for the test to be added to the `resource_board` table (input).

RSRC_UnmaskResources (declaration in `RSRC_defs.h`, definition in `RSRC_rlist.c`)

```
void RSRC_UnmaskResources (int resource_value, *resource_array,  
                           num_to_process)
```

This function unmask a specified number of values out of the masked resource value.

Arguments:

`resource_value`: Masked resource value (input).

`resource_array`: Array to hold unmasked resource values; must be allocated before calling this function (output)

`num_to_process`: Number of values to pull out of the masked resource value (input).

RSRC_UpdateResource (declaration in `RSRC_defs.h`, definition in `RSRC_rlist.c`)

```
void RSRC_UpdateResource (char *name, char *test_id, int value)
```

This function sets the resource to value in table `resource_board` if name is found in the `resource_list`.

Arguments:

`name`: Resource name (input).

`test_id`: Test ID (input).

`value`: Resource value (input).

RSRC_UpdateResourceBoardResourceValue (declaration in `RSRC_defs.h`, definition in `RSRC_dbutils.c`)

```
void RSRC_UpdateResourceBoardResourceValue (char *test_id,  
                                             int rsrc_number,  
                                             int value)
```

This function adds an entry for a specified test to the resource_board table of the Session Manager database.

Arguments:

test_id: Test ID for the test to be updated on the resource_board table (input).

rsrc_number: Number of the resource to be updated (input).

value: Updated value for the resource; acceptable values are RSRC_On, RSRC_Off, or RSRC_Incomplete (input).

set_unique (declaration and definition in tekbase.c)

static int set_unique (int value)

This function .

Argument:

value:

STRNG_IsBlank (declaration in STRNG_defs.h, definition in STRNG_utils.c)

Boolean STRNG_IsBlank (char *str)

This function .

Arguments:

str:

STRNG_Myfgets (declaration in STRNG_defs.h, definition in STRNG_utils.c)

char *STRNG_Myfgets (char *str, int n, register FILE *iop)

This function .

Arguments:

str:

n:

iop:

STRNG_NullOutString (declaration in STRNG_defs.h, definition in STRNG_utils.c)

void STRNG_NullOutString (char *str, int len)

This function .

Arguments:

str:

len:

STRNG_RemoveTrailingSpaces (declaration in STRNG_defs.h, definition in STRNG_utils.c)

void STRNG_RemoveTrailingSpaces (char *str)

This function .

Arguments:

str:

tbl_add_mode (declaration in `tbl.h`, definition in `tbl.c`)

```
void tbl_add_mode (enum tbl_modes mode, accessfun countfun,
                  accessfun getfun, accessfun putfun,
                  accessfun deletfun, accessfun updatefun,
                  accessfun freefun, accessfun donefun)
```

This function associates the specified database access functions with the specified database mode.

Arguments:

mode: The enum value of the mode being defined (input).

countfun: A function which will be called via `tbl_count` to count the number of rows in a table which meet specified conditions (input).

getfun: A function which will be called via `tbl_get` to retrieve data from a table (input).

putfun: A function which will be called via `tbl_put` to add a row of data to a table (input).

deletfun: A function which will be called via `tbl_delete` to remove one or more rows of data to a table (input).

updatefun: A function which will be called via `tbl_update` to change data values in a table (input).

freefun: A function which will be called for each referenced table during `tbl_free_all` to free up any memory allocated by the table access functions and associated with that table (input).

donefun: A function which will be called at the end of `tbl_free_all` to perform any other cleanup when database access is no longer needed (input).

tbl_count (declaration in `tbl.h`, definition in `tbl.c`)

```
int tbl_count (va_alist)
```

This function parses the commands and looks up the named table, then calls a mode-specific function of the form `mcount (table, count, unique, col, conditions, tbl_numconditions)` to count the number of rows of data in the table which meet specified conditions.

Argument:

va_alist: A variable length argument list; the required elements are `tblname (char *)`, `count (int *)`, `is_unique (int)`, `colname (char *)`; the remaining elements are passed to `parse_tbl_commands` as the second argument (input).

Returns FALSE if an error is encountered in parsing the argument list; otherwise the return value is the result of the mode-specific function call.

tbl_delete (declaration in `tbl.h`, definition in `tbl.c`)

```
int tbl_delete (va_alist)
```

This function parses the commands and looks up the named table, then calls a mode-specific function of the form `mdel (table, conditions, tbl_numconditions)` to delete one or more rows of data from the table.

Argument:

va_alist: A variable length argument list; the required element is `tblname (char *)`; the remaining elements are passed to `parse_tbl_commands` as the second argument (input).

Returns FALSE if an error is encountered in parsing the argument list; otherwise the return value is the result of the mode-specific function call.

tbl_free_all (declaration in `tbl.h`, definition in `tbl.c`)
void `tbl_free_all` ()

This function frees any allocated memory and calls the appropriate mode functions to do anything required to terminate table access.

tbl_get (declaration in `tbl.h`, definition in `tbl.c`)
int `tbl_get` (va_alist)

This function parses the commands and looks up the named table, then calls a mode-specific function of the form `mget` (`table`, `first`, `eof`, `unique`, `sortop`, `sortcol`, `commands`, `tbl_numcommands`, `conditions`, `tbl_numconditions`) to retrieve data from the table.

Argument:

va_alist: A variable length argument list; the required elements are `tblname` (char *), `first` (int), `eof` (int *), `unique` (int), `sortop` (int), and `sortcol` (char *); the remaining elements are passed to `parse_tbl_commands` as the second argument (input).

Returns FALSE if an error is encountered in parsing the argument list; otherwise the return value is the result of the mode-specific function call.

tbl_new (declaration in `tbl.h`, definition in `tbl.c`)
int `tbl_new` (va_alist)

This function allocates and defines a new table structure.

Argument:

va_alist: A variable length argument list; the required elements are `intname` (char *), `extname` (char *), `mode` (int), and `DB_name` (char *); the remaining elements, `intcolname` (char *), `extcolname` (char *), and `data_type` (int) are repeated for each column in the table (input).

Returns TRUE if all went well or FALSE if there were any problems.

tbl_print (declaration in `tekbase.h`, definition in `tekbase.c`)
int `tbl_print` (char *tablename, char *testid)

This function .

Arguments:

tablename:

testid:

Returns TRUE if OK or FALSE if an error is detected.

tbl_print_all (declaration and definition in `tekbase.c`)
int `tbl_print_all` (char *tablename)

This function .

Argument:

tablename:

Returns TRUE.

tbl_put (declaration in `tbl.h`, definition in `tbl.c`)
`int tbl_put (va_alist)`

This function parses the commands and looks up the named table, then calls a mode-specific function of the form `mput (table, commands, tbl_numcommands)` to add a row of data to the table.

Argument:

va_alist: A variable length argument list; the required element is `tblname (char *)`; the remaining elements are passed to `parse_tbl_commands` as the second argument (input).

Returns FALSE if an error is encountered in parsing the argument list; otherwise the return value is the result of the mode-specific function call.

tbl_tekbase_init (declaration in `tbl.h`, definition in `tbl.c`)
`int tbl_tekbase_init ()`

This function initializes mode hooks and parameters for TekBase.

tbl_update (declaration in `tbl.h`, definition in `tbl.c`)
`int tbl_update (va_alist)`

This function parses the commands and looks up the named table, then calls a mode-specific function of the form `mupdate (table, commands, tbl_numcommands, conditions, tbl_numconditions)` to update data in the table.

Argument:

va_alist: A variable length argument list; the required element is `tblname (char *)`; the remaining elements are passed to `parse_tbl_commands` as the second argument (input).

Returns FALSE if an error is encountered in parsing the argument list; otherwise the return value is the result of the mode-specific function call.

tekbase_check (declaration in `tekbase.h`, definition in `tekbase.c`)
`int tekbase_check (char *intname)`

This function ensures that the process is connected to TekBase and the database for the current table is open.

Argument:

intname: Internal name of a table (input)

Returns TRUE if OK or FALSE if any problems occur.

tekbase_close (declaration in `tekbase.h`, definition in `tekbase.c`)
`int tekbase_close ()`

This function closes the current TekBase database. It returns TRUE if the close is successful or FALSE if any errors are detected.

tekbase_connect (declaration in `tekbase.h`, definition in `tekbase.c`)

```
int tekbase_connect ()
```

This function attempts to connect to TekBase. It uses the environment variable QYHOST to determine the database server. It returns TRUE if the connection is successful or FALSE otherwise.

tekbase_disconnect (declaration in `tekbase.h`, definition in `tekbase.c`)

```
int tekbase_disconnect ()
```

This function disconnects from TekBase. It always returns TRUE.

tekbase_do_tql (declaration in `tekbase.h`, definition in `tekbase.c`)

```
int tekbase_do_tql (char *tqlstring)
```

This function executes an arbitrary TQL command.

Argument:

tqlstring: The TQL command to execute (input).

Returns TRUE if there are no errors or FALSE if any errors are detected.

tekbase_open (declaration in `tekbase.h`, definition in `tekbase.c`)

```
int tekbase_open (char *DB)
```

This function attempts to open a specified TekBase database. If the program is not currently connected to TekBase, an attempt will be made to connect first.

Argument:

DB: Name of database to be opened (input).

Returns TRUE if the open is successful or FALSE if any errors are detected.

tkbcount (declaration and definition in `tekbase.c`)

```
static int tkbcount (struct tbl_info *tbl, int *count, int unique,  
                    int column, struct tbl_command *conditions,  
                    int numconditions)
```

This function handles execution of `tbl_count` for TekBase tables.

Arguments:

tbl: The current table (input).

count: Resulting count (output).

unique: Whether duplicate values should be counted (input).

column: User column number (1-based) to use for the value-count (input).

conditions: The current list of conditions (input).

numconditions: The number of conditions in the list (input).

Returns TRUE if OK or FALSE if any errors are detected.

tkbdel (declaration and definition in `tekbase.c`)

```
static int tkbdel (struct tbl_info *tbl,
```



```

        struct tbl_command *conditions,
        int numconditions)

```

This function handles execution of the `tbl_delete` function for TekBase tables.

Arguments:

tbl: The current table (input).
 conditions: The current list of conditions (input).
 numconditions: The number of conditions in the list (input).

Returns TRUE if OK or FALSE if any errors are detected.

tkbdone (declaration and definition in `tekbase.c`)

```

    static int tkbdone ()

```

This function closes the TekBase database and disconnects. It always returns TRUE.

tkbfree (declaration and definition in `tekbase.c`)

```

    static int tkbfree (struct tbl_info *tbl)

```

This function is not used with TekBase.

Arguments:

tbl: The current table (input).

Returns TRUE.

tkbget (declaration and definition in `tekbase.c`)

```

    static int tkbget (struct tbl_info *tbl, int first, int *eof,
                      int unique, int sortop, char *sortcol,
                      struct tbl_command *commands, int numcommands,
                      struct tbl_command *conditions,
                      int numconditions)

```

This function handles the one-row-at-a-time retrieval of `tbl_get`, implemented through the use of the TekBase `FETCH...WHERE` command.

Arguments:

tbl: The current table (input).
 first: Boolean indicating if this is the first call for the current query (input).
 eof: Flag set to TRUE when no more data rows are available (output).
 unique: Whether rows retrieved should be unique (input).
 sortop: Whether to sort values, FALSE, TASC, or TDESC (input).
 sortcol: The column to sort on (input).
 commands: The current list of commands (input).
 numcommands: The number of commands in the list (input).
 conditions: The current list of conditions (input).
 numconditions: The number of conditions in the list (input).

Returns TRUE if OK or FALSE if any errors are detected.

tkbput (declaration and definition in tekbase.c)

```
static int tkbput (struct tbl_info *tbl,  
                  struct tbl_command *commands, int numcommands)
```

This function handles execution of the tbl_put function for TekBase tables.

Arguments:

tbl: The current table (input).
commands: The current list of commands (input).
numcommands: The number of commands in the list (input).

Returns TRUE if OK or FALSE if any errors are detected.

tkbupd (declaration and definition in tekbase.c)

```
static int tkbupd (struct tbl_info *tbl,  
                  struct tbl_command *commands, int numcommands,  
                  struct tbl_command *conditions,  
                  int numconditions)
```

This function handles execution of the tbl_update function for TekBase tables.

Arguments:

tbl: The current table (input).
commands: The current list of commands (input).
numcommands: The number of commands in the list (input).
conditions: The current list of conditions (input).
numconditions: The number of conditions in the list (input).

Returns TRUE if OK or FALSE if any errors are detected.

tql_check_types (declaration and definition in tekbase.c)

```
static int tql_check_types (struct tbl_info *tbl,  
                           struct tbl_command *commands,  
                           int numcommands)
```

This function looks up the TekBase types for the commands and sets the global variables TQL_types and num_select_params.

Arguments:

tbl: The current table (input).
commands: The current list of TBL commands (input).
numcommands: The number of commands in the current list, or if this is specified as ALLCOLUMNS, then commands are ignored and the types for all columns in the table are retrieved (input).

Returns TRUE if OK or FALSE if any errors are detected.

SSME Post-Test Diagnostic System Systems Section

**Final Report
Attachment #3**

Transcripts of Interviews With SSME Data Analysts

Erik Sander
4/6/92
Overview of SSME Data Analysis

Org Chart (see attached)

- Chief engineer — Otto Goetz; usually interface with his staff regularly.
 - Structures guys
 - Dynamics Lab
 - Electronics & Controls Lab
 - Materials Lab
 - Propulsion Lab (John McCarty); Propulsion systems; EP51 (Jerry Reitas??); Harlan Pratt — Branch chief of EP52
- Martin Marietta
 - David Vaughan (spending 80% of his time these days with NLS, so don't see a lot of him; chief engineer for Martin's NLS effort).
 - Erik Sander
 - Do everyday data analysis.
 - Jeff Cornelius — AI & data processing guy
 - Rob Smith — dynamics
 - Two Areas:
 - Solid propulsion (Bob Bowman)
 - Liquid propulsion
 - Dave Foust — Flight data
 - Data Analysis (Joe Leahy)
 - Model Analysis (Brian Piekarsky)
 - Data reduction
 - Engine modeling (component & system)
 - Jean Tucker — Model group & AI stuff with Jeff
- Components Branches
 - Turbo
 - Combustion Devices
 - Valves
- Data Review
 - Erik's group responsible for system work; overall interaction of entire machine, valves, etc.
 - Other groups act as specialist groups; they all look at the data too, but in a more focused sense.

Documentation (packages assembled &/or used by Martin guys)

- SSME Data Analysis Handbook
 - What to look for, how to look for it, procedures, etc.
 - (NOTE: I have a copy of this; it primarily consists of the high-level steps to go through in performing an analysis, i.e., what data packages to look at, what analysis programs to run, etc., for different types of tests and for flight).
- Flight Handbook
 - What you typically see on flight, why pressures go up & down, etc.
- ICDs
- LCCs
- Purge Sequences (1,2,3,4, engine ready mode, START)
 - Where purging is done & why, where you see it in the data, etc.
- SSME Orientation Manual
 - Good for schematics & pictures

Analysis Procedures Flow Charts (see attached)

1. Get pretest data (hours to few days prior to test; usu 1/2 to 1 day)

A. Rocketdyne's assessment of the previous test

B. What they're going to do on the next test

-> Analysts obtain from this the following:

- Select comparison tests for data book

Based mainly on hardware; when did it run last, etc. Mainly gives an "eyeballer" to check the next test against (a reference point). What's up, what's down; how the engine changed between the two tests. Usu pretty obvious (e.g., 15th run for same engine on same stand). Want to take possible reasons for change out (i.e., remove as many possible sources of behavior deviation as possible).

- Research hardware history.
- Prepare model input decks with engine characteristics to allow proper reduction of test data.
- Special requests to Boeing for runstream

2. Get phone call: just ran hot test. Info about duration and any FIDs.

Model Group:

- Get 1s avg file from BCCS (or self-made)
- Run data reduction

(Would like to see model stuff integrated with diagnostic system at some point; think it would provide valuable information.)

Analysis Group:

- Wait for ALL data to be plotted (data books) so don't have to wait for each plot needed on request.
- Look for effects of software & hardware changes
e.g., Constants in controller which change MR in engine can effect LOX turbine temps, fuel turbine temps; operation of the whole engine.
- Start looking at data. Usu 1-4 people. One person in charge of each test (Randy responsible for two test stands, Taylor responsible for one).
- Create "Master Observation List" of anomalies (not necc problems; just unexpected observations)- very informal. Don't start explaining them yet, just list them for later analysis.
- Start trying to explain observations. Do I have a hardware or software change which would account for the gain? Most observations are easily explained and not even documented (e.g., if a software constant change accounts entirely for an observation).
- Extent of analysis depends entirely on priority of test and when the program office wants an answer. If it's an important test or something major happened, might have a review only a few hours after the test.
- Person in charge of test stand directs the analysis of any tests on that stand (e.g., might farm work out to Eric).
- Anomaly investigation order a function of two parameters: priority and estimated time to resolve. If an anomaly can be explained very quickly, will often do it first just to get it out of the way. Can usually tell when an anomaly is going to take a lot of time.
- When an anomaly is resolved it is just crossed off the list.

3. Set up data review — Informal meeting with program office. Usually several hours to a day after the test.

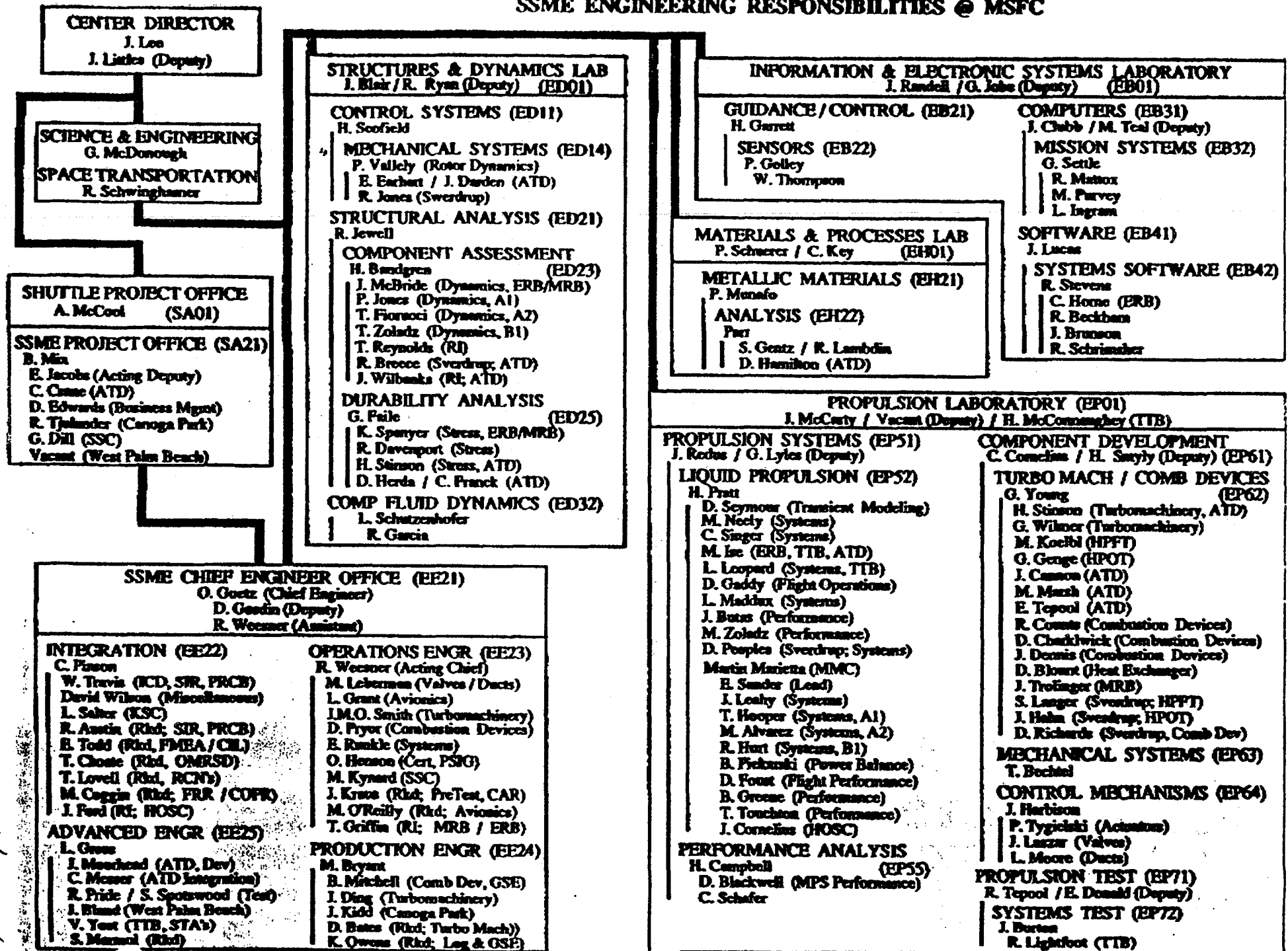
- Example of a major anomaly: Setting at steady-state and fuel turbine temp suddenly jumps up 100 degrees. Severity of anomaly not necessarily a function of the magnitude of the deviations.

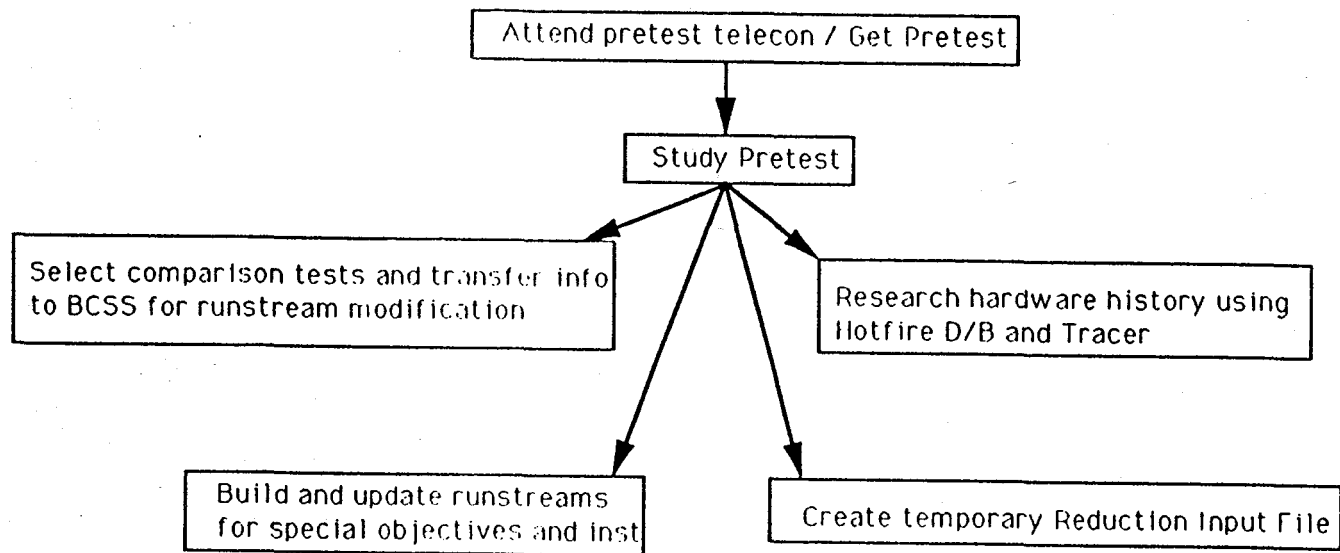
- Spend most of our time trying to determine what's causing what. Trying to find our way back to the root cause of the problem.
- Reason a lot with gains. If this goes up, that should go up, this should go down, etc.
- Usually have larger gains "near" the source of the anomaly (the rest of the system tends to attenuate the effects).
- Probably half of the anomalies resolved before you finish getting through the data package.

Phases of Engine Operation

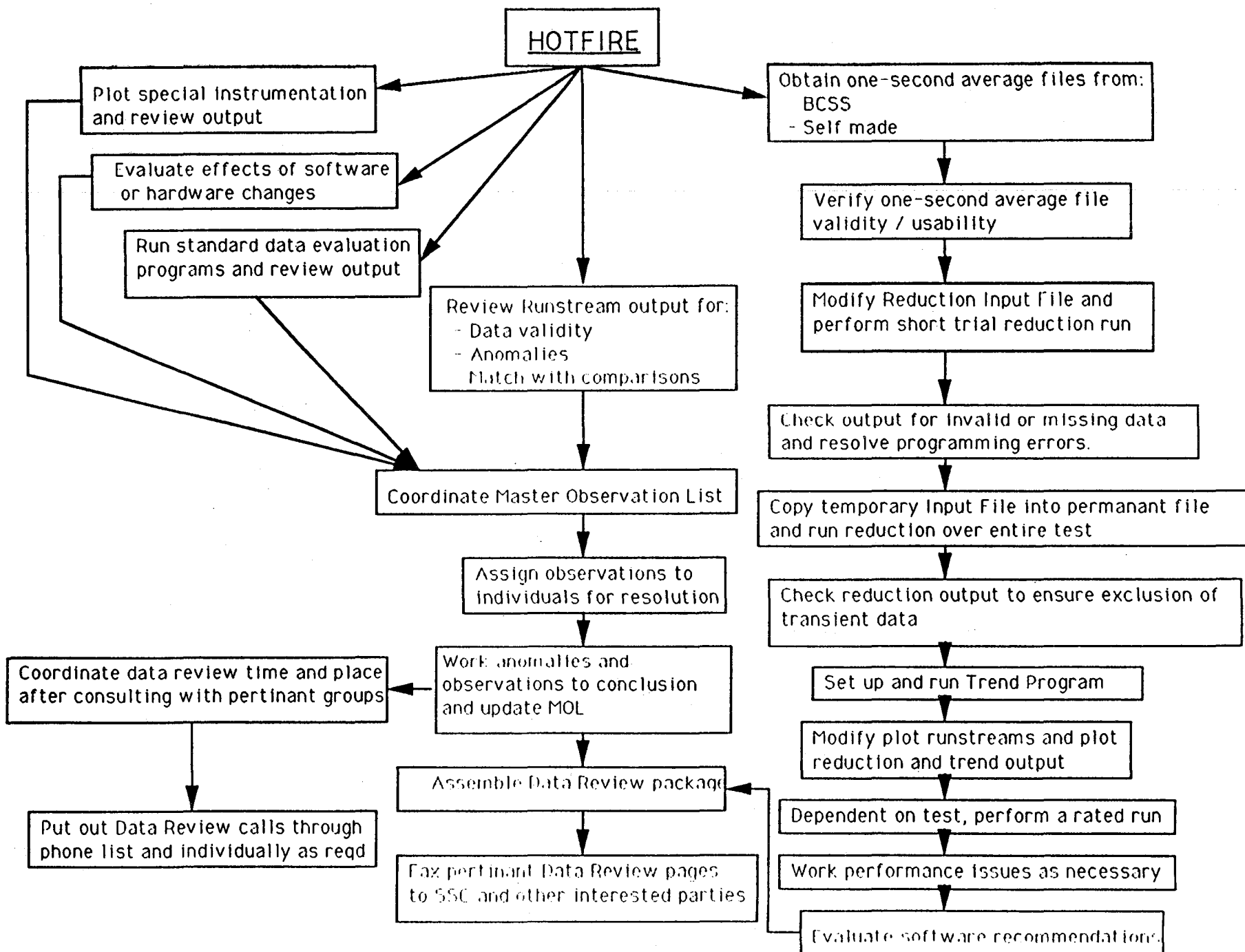
- Phases of engine operation significant to data analysis: pre-start, start, mainstage, shutdown, post-shutdown.
- In all phases the primary goal is to ensure safe operation of the engine, and determine if it is ready for another firing.
- During pre-start also interested in data. What is the facility doing to the engine? Is it properly preparing the engine for firing? Is everything purged properly so that the engine is ready to receive the cryogenics and start in a safe manner?
- Worst thing you can do is blow up an engine. Second worst thing you can do is shut an engine down prematurely, because it becomes dead-weight on the shuttle. Middle ground is to operate in any mode in which the engine is producing some thrust, but at least it's not acting completely like a dead-weight (i.e. lockup).
- Mainstage. Primary Objective is to determine if there is something which would prevent you from running the engine again in a safe manner. Secondarily is to see if the test objectives were met.
- Shutdown objectives. Two different shutdown modes: pneumatic and hydraulic (hydraulic is normal; pneumatic is emergency backup shutdown system). Pneumatic powered by Helium on the shuttle. Also look for "cracked frisbee". Frisbee can get cracks in it allowing LOX to get above it at shutdown, raising turbine temps. Did the engine come down to a stable point? If fuel-side lockup is performed, check to make sure fuel isn't going where it shouldn't.
- Three ways to shut the engine down on test: controller redlines, facility redlines, and observer (e.g., if he sees a fire).

SSME ENGINEERING RESPONSIBILITIES @ MSFC

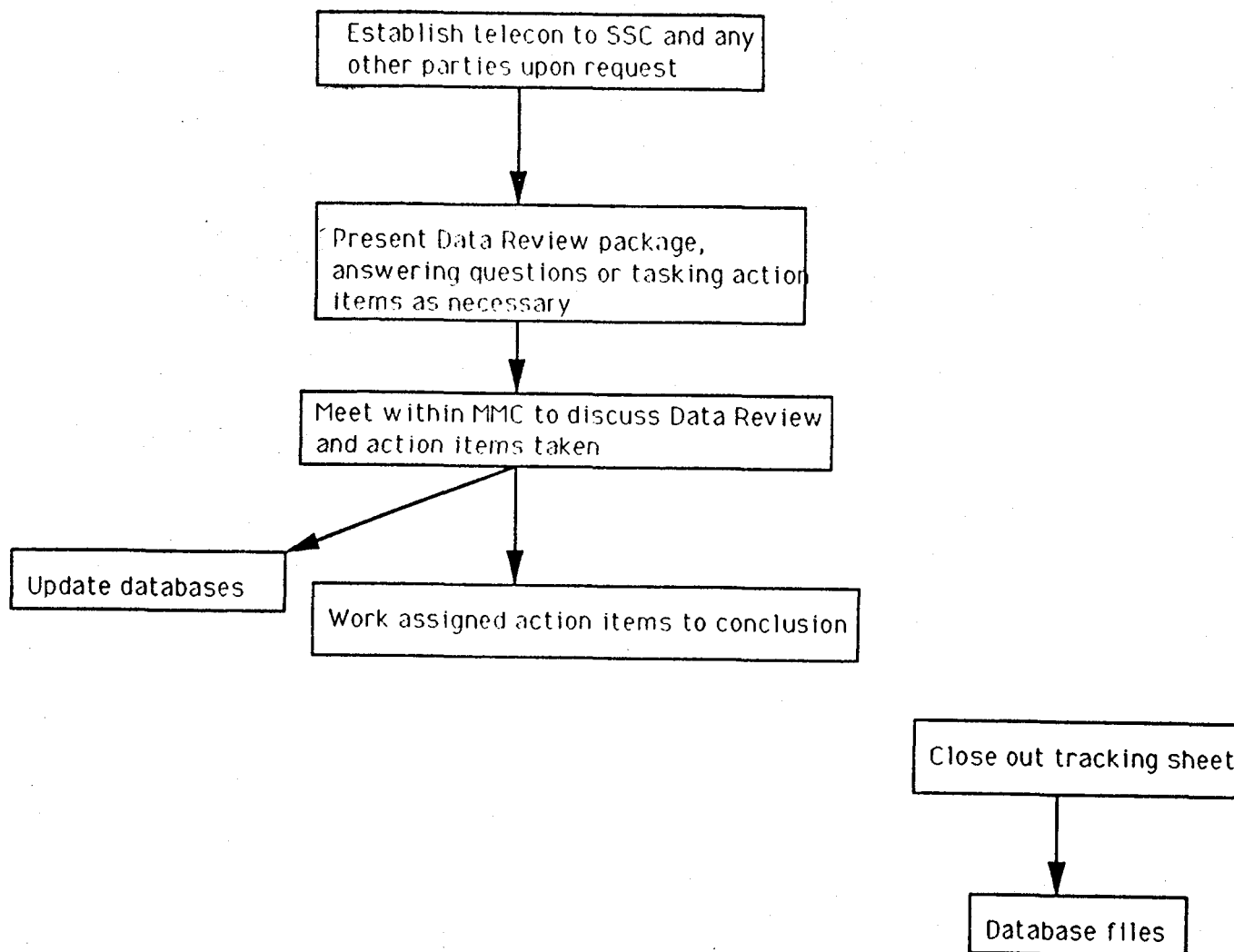




For 11/11/10



For 4/16



Brian Piekarski
4/6/92
Overview of SSME Data Modeling

- Brian is head of models group in Martin; primarily responsible for running power balance (steady-state) model. He has five years experience doing this.
- Models run on an IBM mainframe, run after every test.
- Data from a typical test goes to PE then to IBM.
- In addition to data, have several other input files for model which specify, e.g., about 150 resistances. There is a standard input constants file, and another file which overrides the default input constants (used to tweak the input for a particular engine). Use these files specify unique configurations. For example, have recently been doing development testing on a large throat MCC, which requires us to tweak the model to reflect the change.
- Outputs information about the engine (1350 parameters). Primarily responsible for determining performance. Isp, flowrates, thrust, etc. Give a report on these at every data review. Occasionally there will be some turbomachinery issues, and we will compute efficiencies and head and flow coefficients, etc. These values are all plotted over time.
 - ◊ For Flight engine qualification test: Calculate performance & then compute performance at standard inlet conditions. Will also do this correction if an engine is running way off nominal (to see if the inlet conditions accounted for the abnormality).
 - ◊ There are flow calibration constants used in the controller to achieve a desired MR. If the engine ran off-MR, then the program will compute what the flow constant should be changed to for the next test to get the desired MR.
- Program is about 15,000 lines of FORTRAN.
- Old program. Not well-modularized.
- Two models: Steady-state (prediction and data reduction modules) & Digital transient (does not use test data; just has time-lag coefficients, etc.). Brian is not very familiar with transient model.
- For TTB have tried to compute what some of the internal resistances in the engine are, since we have more data. Resistances are normally an input to the model (there isn't enough instrumentation on the SSME to compute them).
- Occasionally, for new components/configurations, will do some analysis and try to determine what the change in resistances or performance.
- We also trend performance, especially for non-flight hardware which can develop leakages, etc., and have its performance trail off over time.
- Can select time range you want to run the power balance model for. Typically stay 5-10 seconds away from power level changes. Have tried to run it over transient portions (e.g., when we had failures) to see if we can get anything out of the data. We would get some flowrates, etc., knowing that most of the other output is incorrect. There are many iterative convergence loops in the program which might not complete if you ran it on transient data.
- Occasionally use the model as a diagnostic tool. Might be able to verify that there was leak or blockage. Had one engine that had an unusually high pressure/resistance due to a weld bead in the line. When there is a failure, we try to use it as much as possible. However, most failures occur during transients (START or SHUTDOWN) so the model is of limited use.
- Had a failure on B1 several years ago which occurred during steady-state. Typically average the data every second for input to the model, but can run the model on every data point, and might do this when there is an anomaly during steady-state. If there is a turbopump failure, you might see speed drastically changing, and we would run the

model on it as far as possible (when the data starts becoming too transient we can't run the model).

- Can simulate some failure modes. Example: CCV Actuator Case — Ran the model with the CCV valve "closed" (in software) to see if the results agreed with the observations. Also play "what-if" games; what will happen to turbine temps if we have a certain failure. Usually get some things that match the data well, but then usually have a few "oddballs" that don't match (usually due to inexactness in the power balance model). But have used the model to validate or give credability to failure hypotheses.
- Interactions with the data analysts: Close interaction when there is a failure analysis. They screen the data for bad data; analysts will tell us when a sensor is invalid. The model has some screening capability, but not much (fixed reasonableness limits). Sometimes get redundant measurements that don't agree; we work with analysts to determine which one is right (e.g., flow measurements); sometimes they help us, sometimes we help them. It get tricky, because the controller uses the flow measurements to control the engine and if the flow measurement is bad then the effects are difficult to track down. They keep some databases which are useful to us; we keep some which are useful to them.
- At data review, analysts report raw data; we report model results. Pretty much parallel functions.
- We also do flight predictions. Sometimes fly an engine with components which have been tested on other engines, but have never been tested as an assembly. We will predict what the performance of the configuration should be on flight. Time history prediction: for every 10 seconds in the flight.

Databases (See Attached)

- DB of all input files which have been run thru the power balance model. Good second source of data (independent of the Perkin-Elmer; doesn't hold a lot of data on-line). It's not full-sample (it's 1s averaged) but it's better than nothing if you don't have access to the PE. Access is also faster than through the PE.
- DB of all model input files for all ground tests (i.e. with configurations, resistances, etc.).
- DB of all outputs (plots) generated for all ground tests. 1350 parameters per time slice (usu up to 100 time slices). Typically run every 5-10 seconds during steady-state. Can use to quickly plot information about a test 20 tests ago, e.g. to get the previous test for a particular engine. Have statistics programs which can take these in and compute 2sigma curves for statistical comparisons.
- DB of special notes for all reductions. E.g., special changes made to the model for the run, any "funnies" seen in the data, etc.
- DB of more prominent predictions that we do. Predictions run for several normal cases, e.g., different inlet conditions, different resistances, etc. Just keep these around for quick comparisons.
- Establishing a DB of normalized performance for ground tests. Performance for all engines normalized to standard inlet conditions (already done for flight, but to date has not been kept for ground tests).
- Rocketdyne comes out with a new power balance model every couple of years. When they do, we re-run all of our cases through the new model. We have a program setup to automate the updating, so it only takes us about a week to update all of our databases.

Accuracy of Model...

- Does the model respond well to variations in engine configuration (e.g., for development testing of new components)? There are many unknowns in the model. There are many things about the engine which are unknown, e.g., internal flows that aren't measured directly. There are some subtle flow splits that we're not quite sure

of. There are some resistances in there that have never really been measured directly. TTB will give us some insights into some of these models. There are some things that can't be determined even with extra instrumentation, e.g., there are some places in the engine where the temperature varies widely over a small area, and a temperature sensor will only give you the temp at one particular point. Thus, it's hard to tell whether some parts of the model are accurate or not. Some of the resistances are based on somebody's best guess made years ago. We have recently removed the MCC baffles, and we've tried to tweak the model to reflect the change in resistance, etc., but it's real hard to tell how good the modified model is. Some areas of the model are very good, yielding good predictions, and some areas yield results that have to be taken as ballpark figures (I'd trust a random number generator as much as I'd trust some parts of the model). I've been working with the model for five years, and it takes that long to know what parts of the model to trust.

Tell me About the Reduction Model...

- Combination of physical relationships and "fill-in-the-blanks" empirical relations (e.g., curves from data).
- Model is very poorly documented.
- Model was originally developed by two guys at Rocketdyne, one of which has died. The other developer is now a very senior engineer who says that he has never liked documentation and that he has no plans to document the code. There are now some younger guys at Rocketdyne working on the code (Brian Davis?) who have been trying to document it, but they don't know where a lot of the relationships have come from. They have put together a user's manual. A lot of the documentation that we have, we wrote. Most of the variables in the one large input array ("A" array) are reasonably well documented.
- It's hard to trace where a value comes from in the program. There are lots of subroutines, data tables, and lookup curves which are completely undocumented and have parameter names like "x". Some of the Rocketdyne comments are of the form: "The following line is bogus.", so don't know whether to leave it in or take it out. Also: "The following code needs to be changed, but other things need to be changed first, so we'll just leave it like this for now."
- It was our job to get up to speed on this program, and we have only learned how it works after years of experience. If I were to leave it would be a tremendous hit, because I have a lot of unique knowledge about how the model works that is currently undocumented.
- We get a lot of questions about gains (both groups get a lot of these questions). E.g., if we change the inlet pressure, what are the effects on the engine? Rarely is there just one thing changed on an engine between firings, so it makes it more difficult (e.g., multiple hardware changes, changing inlet conditions, etc.). Questions about efficiencies only we can't answer because they are computed values (data guys can't answer them). E.g., If I change the pump efficiency by 4% what will it do to the turbine temps? They're still playing with the Pratt pumps, so get questions like this a lot lately.
- Occasionally get asked questions that are kind of tricky, that the model is not set up to directly compute. E.g., to simulate a valve position (like the CCV failure), you have to go in and hardcode the resistance across the valve rather than directly specifying the position.

What tools would be useful to you?

- Something to pick out anomalous values in the data. E.g., something to be a pre-check on the data before we send it into the model. Sensor validation.

EP / MARTIN MARIETTA SSME DATABASES (continued)

<u>DATABASE</u>	<u>PRIMARY USER</u>	<u>FORMAT</u>	<u>DESCRIPTION</u>
1 Second Avg Data	Model Group	IBM 3084	1 second averaged data files for ground tests
Model Input Files	Model Group	IBM 3084	Input files for Phase II ground tests
Tracking Sheets	Model Group	Paper Files	Model information for Phase II ground tests
Model Output Files	Model Group	IBM 3084	Data reduction output files for Phase II ground tests. Performance and Turbomachinery databases are generated from this data using special access routines
Power Balance Database	B. Pickarski	IBM 3084	Power Balance prediction output files at various operating conditions
Test Summary Database	B. Pickarski	Word	Engine configuration, hardware units, and power level information for Phase II tests
Thrust Profile Database	B. Pickarski	Paper Files	Thrust profiles sorted by standard test profile, power level, and/or propellant vents
E8 Test Data	J. Leahy	IBM 3084	P&W HPFTP and HPOTP component level test data files conducted on the E8 facility
E8 Model Output Files	J. Leahy	IBM 3084	Data reduction output files for P&W HPFTP and HPOTP component level tests
Start & Cutoff Pop Database	?	Symphony	FPB, OPB and MCC Pops

EP/AM/4/11
24-4

EP / MARTIN MARIETTA SSME DATABASES

<u>DATABASE</u>	<u>PRIMARY USER</u>	<u>FORMAT</u>	<u>DESCRIPTION</u>
Anomaly Database	Data Group	Rbase	Key information of anomalies for SSC ground tests
2 Sigma Database	Data Group	Symphony	Statistical data for key engine parameters at certain operating conditions for ground tests and flights.
Primetime Database	Data Group	Symphony	FPB, MCC and OPB prime times for ground tests
Phase II+ Database	Data Group	Cricket Graph	Engine 0209 and 0215 Phase II+ special instrumentation data
Hex Temp Database	D. Foust	Cricket Graph	HPOT discharge temps vs. Hex i/f temps for all flight engines (post 51L)
Flight Changes Database	D. Foust	Rbase	Hardware and software changes for all flight engines (post 51L)
Flight Prediction Database	D. Foust	Symphony/ Cricket Graph	Rocketdyne and MSFC flight predictions, flight data, and deltas at engine start +200 seconds for all flights (post 51L)
Model Flight Database	D. Foust	IBM 3084	Flight profile model prediction output files and flight data plot files for all flights (post 51L)
Green Run Database	Data Group	Symphony	Statistical data on parameters required for hardware to pass green run requirements for flight
Hotfire Database	Data Group	Rbase	Engine hardware information for all ground tests and flights
Pressure Drop Database	M. Alvarez	Symphony	Pressure drops across engine components
Hardware Database	Data Group/ Combustion Devices	Rbase	Post test hardware inspection reports for all ground tests
Early Cutoff Database	L. Maddox	Reflex	Information and paperwork for early engine cutoff tests
DCU-AB Database	D. Gaddy	PC-Word	Engine responses to DCU-A halts

Revised 02-5

Erik Sander & Randy Hurt
4/7/92
Pre-Test Package

Erik Sander...

- Make a folder for each test. Has all information about the test. Includes review presentation packages from our group and other groups. Includes pre-test package.
- Data review: Systems group goes first — with data analysis guys. Next model group. Next dynamics. Next strain gage guys. Then turbomachinery. Then combustion devices. Each group's package goes into the folder for the test. (Note: These folders kept in filing cabinets in Martin's office area.)
- If there is a failure, will often do a separate package on the failure itself.

Pre-Test Package

- A1693 test example. Engine 0219 (has been on the stand for awhile; 7-9 tests). Was the last test ran on A1. Redesigned powerhead engine. Certification testing of the powerhead and hydraulic lockup test (have been doing a lot these lately).
- Hydraulic lockup test — Pull the hydraulics from the engine and see how it reacts.
- Get the pre-test about a day before the test.
- Usually go over the package via telecon with Rocketdyne and Stennis.

First 1/2 of the pretest covers Rocketdyne's analysis of the last test on this stand.

- We've already covered all of this analysis, so it's not really any new information. If we are not switching engines on the stand, then we cover this in detail. At this point we've already told the program office what we think about the previous test, and this is a chance for us to compare notes and iron out any differences that exist.
 - ◊ Summary of Results. Objectives and how they were met.
- Two definitions of engine data analysis. 1. Looking at the current test in isolation. 2. Looking at the performance of the family of engines over time. (Analysis of the program.) If analysis of the program: the two tests need to be related in some way. Typically trying to build off the results off of one test to build the objectives for the next series of tests. Think you would want to initially build the system to do #1 (look at test in isolation).
- To understand the objectives for the upcoming test, you must go back to the previous test and its objectives and how they were met.
- Main things that feed over from one test to another: What instrumentation failed on the last test, so we know to keep a special eye on it for the next test (sometimes Stennis doesn't get things fixed from one test to the next). Sometimes you have a piece of instrumentation that's "hanging out in the breeze". For example, on this stand for this series of tests, have a fuel preburner pressure that's reading ambient because they need its port for an accelerometer. There's only a certain number of bosses on the engine available for instrumentation. That's an example of something you'd expect to be bad on the next test.
- If you see an anomaly on one test, will you still be looking for it on the next, even if it's supposedly been fixed? Yes. Example: piston ring seal; A round seal that sits around the top of the MCC. Sometimes during a test it will shift a little. Will rebalance the fuel system on the engine just slightly. Steps: We see certain things in the fuel system shift, look at directions and magnitudes. E.g., fuel pump speed changes, pressures in that leg shift, and the temps shift slightly. We come to associate these shifts with a piston ring seal shift. We did analysis on it once the first time we saw it, now we just know what it is. If the effects are seen again, the analyst should immediately know that it's a piston ring seal shift, and just have to determine the effects of that on the engine. If seen on one test, we will probably be looking for that again on

the next test, and maybe the test after that. The seal could continue shifting to the point at which it becomes a problem.

- Example on this engine: On previous redesigned powerhead engines we've seen accelerated MCC degradation (it cracks terribly). These chambers cost \$4M and take 4yrs to make, and we don't have any spares, so it's a very important item, it can be a "show stopper" on a program. These redesigned powerheads in the past have just "eaten up" MCCs (cracked them); we usually get 40,000 sec of life, now we're getting 5,000 seconds. So, if we see evidence of increased cracking on a test, esp. with a new powerhead, we will always be on the lookout for further cracking for the engine. It will be one of the first things we look for since it is such a high priority.
- Usually recognize most major anomalies within the first hour of getting the data.
- You'll see all kinds of shifts and movements in the data. How fast you can determine what's a normal reaction and what's an anomaly comes down to experience. For the diagnostic system that translates into two things: gains and databases.
- Also in pre-test: comparisons to program history relevant to current objectives. Comparisons of just with the last test, but with the last 10 tests. How is the program progressing overall?
- Rocketdyne doesn't break up their data packages (pre-test) in the same way we do (post-test). They do in their internal reviews, but in the reviews with us they combine highlights from all of their different disciplines.
- After the test, technicians will measure things like torque checks and shaft travel. Information about these tests are also in the package. Most of this is done on the pumps. Very often we come up with theories about what kind of anomaly occurred, but it isn't until the post-test inspection that these theories get confirmed or denied. Example: we see one of the pumps coming down very quickly on shutdown (i.e. very high deceleration rate), and hypothesize that the pump is "locked up". You then wait for the post-test inspection torque test to prove or disprove this. Means you probably did some damage internal to the pump, e.g., to the bearings. At this point you might go back to the data and see if you can find more clues about exactly what happened within the pump. Would work very closely with turbo guys to resolve this.
- In the systems group, you don't get knocked out of any problem analysis. By definition you never have a system problem; it's always a component that fails. So you have to work closely with the component specialists.
- For each anomaly on the previous test, will list effects (e.g., for cracking what is the effect in terms of increase leakage?), and actions taken to correct the anomaly (sometimes none if minor and there is a tight schedule).
- UCRs simply note anomalies; they are not things that must be fixed prior to next test (??). NASA control over what must be fixed is exercised when they sign off on the next test.
- Analysts don't get involved in filling out paperwork such as UCRs; program office handles that.

Second 1/2 of Pre-Test Package

- What we're going to do on the next test.
- Where we get most of our information on what's going to be done. This is critical information to us (possible problem getting this into exper system). Most of this info is in TRACER.
- First thing we'll go thru is the objectives. Some of these never change.
- Example objectives: Pneumatic shutdown with hydraulic lockup. Evaluate valve drift due to leakage in actuators (have seen before, but not supposed to happen). What we look for: what system effects would we expect to see as a result of this? E.g. What's the effect of hydraulic lockup? Well, the valve drifts and the engine performance is probably going to change.

What's the effect of CCV cutoff ramp? What phases of operation are going to be affected? Make sure we get right information ready for post-test analysis.

- Thrust profile: Very important. Tells you three things. 1. What power levels you expect to see. 2. vent and pressurization schedule. 3. Propellant transfer.
- During a typical flight, you have a certain amount of cryogenic pressure at the engine inlet plane. Have LOX and fuel sitting at low pressure pump inlets. Temp will remain relatively constant throughout the test. Pressure will change drastically. Pressure made up of 3 things:
 - ◊ 1. Amount of head from the weight of LOX in tank.
 - ◊ 2. Acceleration of vehicle (biggest change seen when solids are jettisoned, change from 3G to 1G and causes major shift in engine behavior).
 - ◊ 3. Ullage pressure in tanks. Engine responds greatly to changes in these inlet pressures, because if pressure is high the pumps don't have to work as hard, that translates into lower turbine temps and different engine operation. On ground test, we try to simulate these differences in engine inlet pressure through venting. E.g. starting with a nominal LOX pressure of 70psi, we will drop it to some lower point; this is called venting. We will then bring the pressure back up by pressurizing the ullage with Nitrogen(?); we call this pressurization.
- Analyst will know what changes he expects to see in the engine (e.g. in turbine temps) as a result of the scheduled venting and repress. Venting and repress is manually controlled (via a guy on a valve), so the timing is not exact.
- In general, everything you do on flight is much more consistent than on the ground. The profile is exact, and you do it over and over again. On the ground you change the profiles, the venting and pressurization, etc.
- Can definitively tell venting and pressurization from data via sensors in ullage, sensors at engine inlet, and sensors at bottom of LOX and fuel tanks (???).
- Pressurization is different than Repressurization (repress). Repress is the gas flow from the engine to the propellant tanks to keep them pressurized during flight.
- Propellant transfer : TTB and Stennis propellant tanks only hold enough for about 300seconds of firing. For longer tests, transfer propellant from barges through line into tanks during the test. See negligible effects on pressure, see good effect on temperature. LOX and Hydrogen (esp hydrogen) heat up as they go through the transfer line. They only cause about 1/2 degree difference, but that is significant (the engine will react to it).
- On this test we know they're using an FPOV valve which leaks terribly (its actually outside of specs), and we're doing a hydraulic lockup, so we expect to see significant valve drift. Effects we expect to see: FPOV will close, which is the controlling valve to the fuel side, so the fuel side will power down, the LOX side will power up (not due to controller, but due to rebalanced flows in system).
- Gains very important. If valve closes by this much, how much should turbine temps change by?
- Pneumatic shutdown: We're going to shut the engine down with Helium instead of hydraulics.
- Get all engine inlet conditions from pre-test package. Helium and nitrogen pressures, temperatures, propellant pressures and temperatures, etc. These are all things that we're going to "do" to the engine.
- Hardware Changes — Very important section for analysts. If a new piece of hardware is going to be put on the engine, we will pull the data from the last time that component ran and compare it with the component being swapped out to try and estimate what the effects on engine behavior should be. E.g., if you get a dog pump, we're going to run very hot. If the pump has always run hot on previous tests, and you see it running hot on the next test, that

would cause it to go from being an anomaly to being an observation (because we understand it).

- Software configuration changes — Also important. Two software constants of primary interest: C2 and Kf (fixed for a given firing). C2 sets the fuel flowrate that you're going to run at a given power level, we use that to set the MR of the engine (want at about 6.01). (Note: Erik has a plot somewhere of C2 vs. resulting fuel flow for a given power level.) Kf: There is a fuel flowmeter between the LPFP and HPOTP. We compute fuel mass flow from this flowmeter (volumetric flow) and the pressure and temperature of the fuel. Translation from the speed of the flowmeter to volumetric flow is a function of the efficiency of the flowmeter. Kf is the calibration constant for the flowmeter which adjusts for differences in efficiency.
◊ Want to control the engine to a given MR. During firing, things will happen which cause MR to change. E.g., cracking in the nozzle or the MCC which takes some of the fuel going into the engine and throws it overboard. This causes the LOX side to power up to give the same MCC Pc, but with less fuel. More LOX, same amount of fuel gives higher MR. We might continue to test the engine in this condition as long as things stay within certain tolerances, but once the tolerances are exceeded we'll change C2 to bring MR back into where it should be. Now will have the same MR, but with more propellants.
- Instrumentation changes — Won't change the way the engine operates, but changes what we see. Of particular note is what additional or fewer parameters we expect to have on the test. We don't worry too much about strain gages; we worry about pressures, temperatures, and speeds.
- Facility changes.

Randy Hurt Takes Over...

- Hardware changes are the most important thing we get out of the pre-test package.
- Changes in pumps are very important.
- E.g. There's a difference between Inconel and Titanium high-pressure fuel duct (different diameters cause different static pressures and discharge temperatures).
- Most of this are notes that you just want to be aware of. Changes in redlines, etc.
- Facility changes usu won't cause changes in engine behavior. Facility fuel flowmeter change can be significant; we use it to cross-check the engine flowmeter.
- Mainly just want to note things that could affect engine performance so you're aware of it.
- Turbine discharge temp redlines change depending on the fuel pump, and if a green run test or not. If first run of a pump might set turbine discharge temp redline at 1660, otherwise might set it at 1760. If the pump has been run before then it's either been rejected for flight, or has come back from flight, and is being used for development. For a greenrun test you don't want to stress the pump too much; you want early problem detection so they can fix it.
- Redlines don't really change much from test to test.
- If we've changed instrumentation but haven't changed scaling constants in software, I would make a note of that.
- Facility redlines will change based on what's being tested. E.g. might change tolerances on temps within a turbine to look for ball bearing problems.
- HPFP speed redline. Used for development engine or first run of the pump. If exceeded indicates you are leaking too much fuel (e.g., nozzle leak) and your pump has "gone away".

- There are also redlines on facility temperature sensors mounted external to the engine, used to detect fires.
- CADS computer simulates what the vehicle would do to the engine (as a controller).
- OPOV Limit Command: Limits how far the controller can tell the OPOV to open. (Note: This appeared on the CADS sheet; this may be dynamically set by the vehicle controller during flight??).

Immediately Following Test...

- Stennis will call to say if the test ran to scheduled duration, and if there were any FIDs.
- Stennis will FAX a quick-look sheet (see attached). Any FIDs would be listed on the top of the sheet by number. We'd have to look the FID up in the software spec to see what actually happened.

SSME # 2107

SSME QUICK LOOK DATA

TEST # 902-555DATE 4/8/92 TIME 11:38 DURATION: SCHEDULED 260 SECONDSACTUAL 259.92 SECONDSFIDS AND/OR COMMENTS NONE!

PRED	109	1470	1470	1780	1800	—	67	81	—	—
TIME (sec)	%RPL	HPOT DT (R) A B		HPFT DT (R) A B		MCC PC (psia)	OPOV (%)	FPOV (%)	LPFP PD (psia)	
20	104	1394	1403	1650	1721	3130	64.1	77.9	252	
110	104	1371	1384	1644	1734	3124	65.0	78.1	243	
210	109	1434	1453	1704	1768	3276	68.5	80.1	254	

FUEL NPSP	
TIME	NPSP
10	23.20
100	7.27
—	—
—	—

LOX NPSP	
TIME	NPSP

MIXTURE RATIO	
TIME	M/R
20	6.00
90	6.02
180	6.03
250	6.02

RASCOS		
TIME	PRP	HPFP
50	1,1,1,2	4,3,3,2
100	1,1,1,2	4,2,2,2
150	1,1,1,1.6	4,3,3,2
225	2,1,1,1.40	4,3,2

4/8/92
4/8/92

Randy Hurt
4/7/92
"Gains"

- Everything in the engine is interconnected.
- The low pressure pumps are driven by the high pressure pumps.
- The preburner MRs are dictated by each other somewhat. E.g. Ox preburner MR goes up, causing pressure and temperature to go up in preburner. That increases the resistance in fuel line going to the OPB, thus causing more of the fuel flow to be routed to the FPB, where the MR then decreases. Thus the Ox side "powers up" and fuel side "powers down".
- Pump change-out gains very important. What happens when you put in a less efficient pump?
- Pump efficiency is indicated by the temperature increase across the pump. Inefficient pump will cause more of a temperature rise, because you're heating up the fluid more.
- Pretty much assume line resistances are constant, and don't really know any of the resistance numbers.

HPFTP

- Fuel mass flow is constant for a given power level.
- Speed is a function of volumetric flow and efficiency. If efficiency goes up, speed goes down for the same flow.
- A large percentage change in LOX pump efficiency will not affect the fuel side as much as a large percentage change in fuel pump efficiency, because the fuel pump is larger, has a larger preburner, and is getting roughly twice the amount of fuel as the OPB. We saw this with the Pratt ATD fuel pump, had a larger turbine area so didn't need as much FPB pressure, and it caused the fuel turbine temps to drop. This lowered the resistance for the fuel line going into the FPB, which caused more fuel to be directed to the OPB causing its MR and combustion temperature to go down. This also lowered the required discharge pressure for the HPFP. Thus pump discharge pressure is a function of turbine efficiency (inversely related).
- We don't have a good measure of turbine efficiency, because we don't have the sensors to get the change in temperature across the turbines.
- A nozzle leak will also affect HPFP discharge pressures. Leak causes downstream resistance to go down, thus causing pump discharge to go down.
- HPFP inlet pressure and temp affected by LPFP. But I don't usually look at these much. LPFP goes from 40psi to about 250psi. This change in pressure doesn't cause large temperature fluctuations. Even with pump changeout you won't see much affect on the large pumps, or on the overall engine performance.
- HPFP — Discharge pressure affected mostly by downstream resistance and not so much by changes in inlet pressure, even if there were large fluctuations in fuel inlet conditions. You might see its speed go down, or the outlet temperature go down.
- HPFP outlet temp — Strictly a function of pump efficiency (inlet temp doesn't change much). If efficiency goes down outlet temp goes up (with same discharge duct).
- HPFP speed — Big pressure increase on inlet pressure would cause this to go down; but this rarely happens (would have to be 20psi or more). Speed mostly a function of pump efficiency and discharge pressure required. If downstream resistance goes down, speed would go down.
- HPFTP — Don't usually think about turbine efficiency because we can't measure it; usually think of the turbopump as a unit. If the turbine efficiency went down would have to have more FPB pressure, will affect downstream resistance of HPFP, will require higher HPFP discharge pressure, which demands more power, so it bootstraps. So for a small turbine efficiency change, will see a large change in pump discharge pressure. If turbine efficiency goes up, less HPFP discharge pressure, less FPB PC, FPB MR down, colder turbine discharge temps.
- HPFTP discharge pressure — measure hot gas injection pressure (from HPFTP to MCC), pretty much dictated by PC. Have a known resistance across the injector and the faceplate.
- HPFTP discharge temp — Based on FPB MR; if MR goes up, temp goes up. MCC MR stays pretty constant, preburner MR can change. If efficiency goes down, turbine discharge temps go up, because you need more pressure in the preburner, causing less fuel flow to the preburner, causing MR to go up.

- If efficiency of both high pressure pumps went down, would require more preburner pressure on both, causing the fuel line resistance downstream of the HPFP to go up, increasing the required HPFP discharge pressure, dropping the fuel flow to both preburners, causing both preburner MRs to go up and their temperatures to go up, and the resistance would cause a proportionally larger amount of fuel to be run through the MCC coolant circuit.
- Changes in HPFTP efficiency effect the engine much more than HPOTP efficiency changes, because it is a bigger pump. If HPFTP efficiency went down, have to open up the FPOV, and will definitely have to open up OPOV. (Efficiency of pump AND turbine considered as a unit. However, changes in efficiency usually due to turbine efficiency change, because you have turbine blades which have much more heat transfer on them than in the pump.)
- If the liftoff seal closes down, ???

FPB

- Don't have any pressures or temps on the fuel inlet. These are affected more BY the preburner, then they affect the preburner. OPB combustion controlled with LOX flow via OPOV (since we're running so fuel rich).

OPOV, FPOV

- Available pressure in OPB LOX inlet line will affect OPOV position, but that's about it.
- If you don't have as efficient a preburner LOX pump, will cause OPOV and FPOV to open. So preburner pump efficiency can be driver in preburner valve positions.
- If you had a very efficient HPFP, you wouldn't need very much LOX in the OPB, so you'd close down FPOV, which would increase the available pressure at OPOV, so you'd close it down too.
- If you increased efficiency on fuel side, decreased(?) efficiency on LOX side, would cause FPOV to close and OPOV to open(?).
- If you decrease LOX tank pressure, then have to speed up HPOP, have to open up OPOV. But as you speed up you get increased pressure, so you need to close OPOV down. So it's a question of which needs more. FPOV stays about the same. So during venting and pressurization, OPOV controls the engine reaction, and FPOV stays about the same.

MCC

- No direct measure of combustion temperature. Have indirect measure via MCC coolant discharge temperature. Temp should be pretty constant. MCC coolant discharge temperature changes are more a function of the amount of fuel flow through the coolant circuit. If this flow increases, the coolant discharge temp should go down.
- If the MCC has a crack in it, then we are leaking fuel out. That causes more cooling in MCC (via film cooling), so the coolant flow should come out colder(?).
- Piston ring seals. There is one on the top and one on the bottom of the MCC. If these shift, they can change the resistance downstream of the HPFTurbine. If the resistance goes down (due to a leak through the seal), the HPFTP speed goes up(?).

HPOP

- Inlet pressure can go up due to tank pressurization. Don't have as much delta-P required, so pump speed goes down.
- Discharge pressure for a given engine will always be about the same, unless you broken a bunch a posts or something like that. The discharge just needs go through the MOV and then into the MCC.
- Discharge temp not measured. But, would be a function of efficiency of the pump. Decrease in efficiency would cause a discharge temp increase.
- Speed mostly a function of inlet pressure. If inlet pressure goes up, speed goes down.
- Discharge pressure is constant. LPOP is driven by HPOP outlet, so inlet pressure of the HPOP is always about the same(?). LOX inlet pressure to LOX turbine relatively constant(?).

HPOT

- Turbine outlet pressure, pretty constant. Because of hot gas pressure requirements, which are pretty constant (commanded by PC).
- Turbine outlet temp. Depends on efficiency of the pump, efficiency of HPFTP, OPB MR. If HPFTP efficiency goes down, it requires more pressure in FPB, raising the resistance in that leg of the fuel inlet, forcing more fuel into the OPB, causing its MR to go down, causing the HPOT discharge temp to go down. Driven directly by MR.

LPOP

- Delta-P across LPOP stays relatively constant. So, changes in engine inlet LOX pressure are passed on 1-for-1 to the HPOP inlet (e.g., if inlet LOX goes up 50psi, HPOP inlet will go up 50psi). So the HPOP must make up for any changes in inlet LOX pressure. So have power-up or power-down the LOX side of the engine depending on the inlet conditions.
- Outlet temp is measured and is relatively constant, due to fairly constant amount of energy into the pump.
- Speed. Don't look at it very much, but it's pretty constant because of constant delta-P.
- LPOP turbine inlet pressure doesn't change much, because it's tapped off of HPOP discharge, which doesn't change much.
- Venting and pressurization have much larger effect on the engine than on fuel side, due to higher density. Start with 160psi head, down to 30psi during vent, then back up 160psi after pressurization.

LPFP

- Outlet pressure doesn't vary much. Varies some with inlet pressure (30psi down to about 5psi). Pumping constant delta-P, like on LPOP. Main driver: see about a 300psi change when you start repressurization.
- Outlet temp pretty constant, but a function of inlet temp. See this change a little with fuel propellant transfer (fuel tends to warm up a little).
- Speed. When repress goes from min to max, the LPFP speed changes. When repress opens, you have less pressure downstream, giving a larger delta-P requirement, so you get more speed.
- Turbine discharge pressure mainly a function of repressurization.
- Turbine discharge temperature pretty constant, but mainly a function of MCC coolant discharge temperature.
- If repress flow is increased, have bigger delta-P for pump, it speeds up, turbine discharge pressure drops, turbine discharge temp down(?).

Preburner Pump

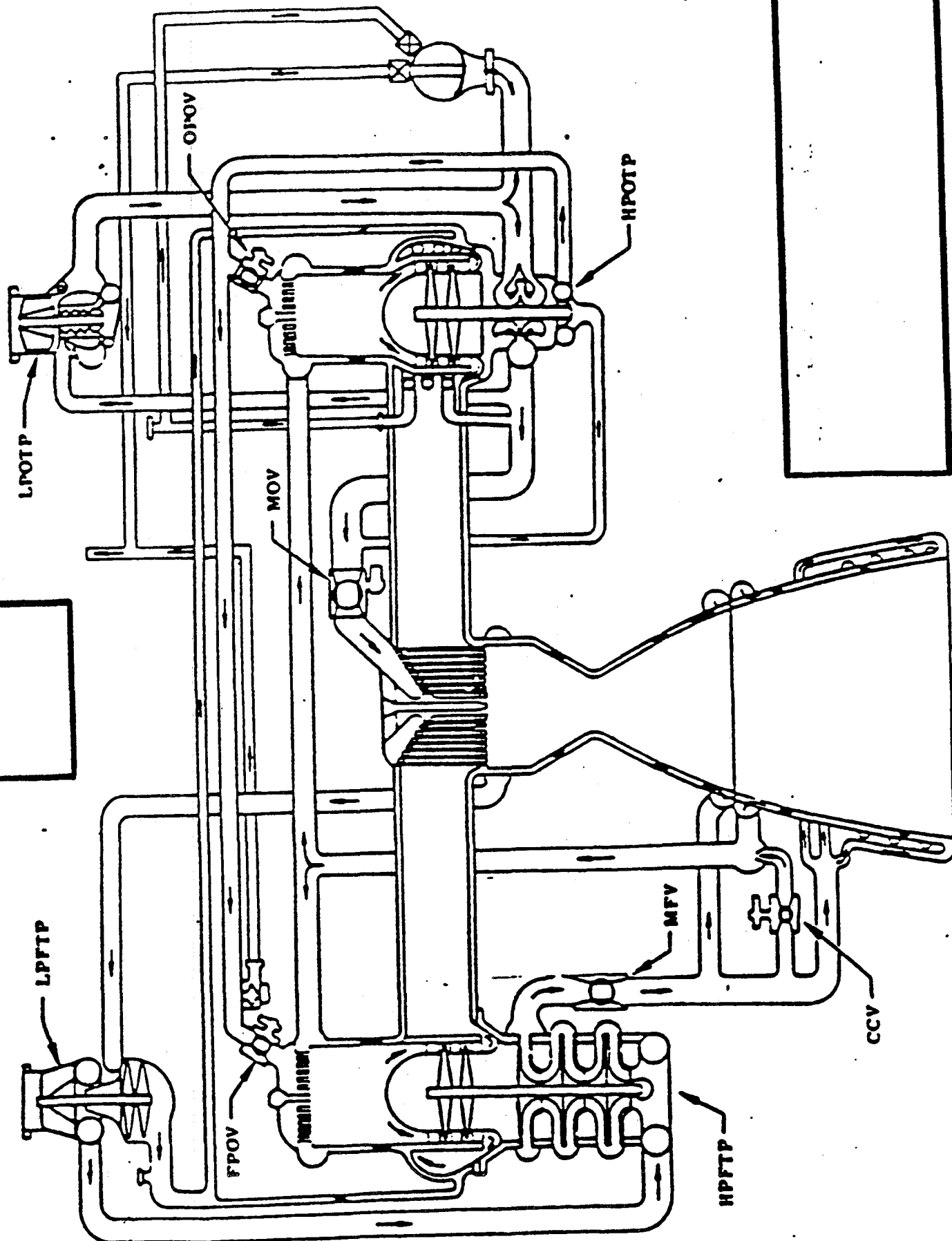
- When you vent, you have a lower LOX inlet pressure, lower inlet pressure to HPOP, HPOP has to speed up, HPOP has same discharge pressure, so preburner pump discharge pressure increases. (Speed causes PBP delta-P to increase, but have same inlet pressure.) This causes OPOV and FPOV to close down.

Heat Exchanger

- Tap off HPOP discharge, vaporize it, send some of it to LOX repress, some of it to POGO repress.
- HX interface temperature (of GOX to tank), required to be within a certain range. Changes according to turbine temp. If HPOTP discharge temp goes up, HX interface temp and pressure go up.

CCV

- Throttles with power level below 100%. Always does the same thing, so don't look at it too much.



Randy Hurt
4/7/92
Post-Test Review Package

Randy walked me through an analysis of the plots in a post-test review package from an old test (A2552). This was not the full data book; just a summary of the more important plots, "enough to satisfy the chief engineer that everything is OK."

- Used to do 1.5sec, 300sec, 550sec tests, now just do 300 and 550sec tests.
- Green Run. Usually get an engine that's going to on the shuttle to fly. It will have a couple of pumps that need to have a green run (pumps only get one green run). The engine itself needs to be calibrated (C2 and Kf in the controller), so it will get two tests before it flies; a calibration test and an acceptance test. Orifices are also installed to tune the engine: there is one orifice at the LPFP. Green run lasts 300sec. Hopefully, MR was pretty close to where it should be.
- Acceptance Test. Typically change the pumps out again to new pumps which need to have green runs, and run the acceptance test for the engine.

Package

- Cover page: Programmatic info and test results. Results are almost always "satisfactory".
- Instrumentation Observations: Noted a 12psi delta in MCC Pc. Not terribly significant, but MCC Pc very important. Have had some clogging in the sensor which causes the bias. The problem is that if the clogging gets worse, the bias goes up exponentially. If you get more than a 400psi delta in MCC Pc you go into electrical lockup.
- Facility fuel repress valve timing (see Fig 1).
- Main info out of pre-test: Hardware changes, max min repress, objectives.
- FFM slope (fuel flowmeter slope), used to computer/evaluate Kf. Kf is calibration constant for the fuel flowmeter. Want it mostly calibrated at 104% since that's where you're running the most.
- Cold flow OPOV and FPOV in order to set open-loop timing parameters (used during START). See Figure 2.

Start

- MCC Pc compared to previous three tests of this engine. Would expect these overlay almost exactly. Might change a little engine-to-engine. OPOV and FPOV really come into play; most of the other valves are pretty much "straight up" open. Expect to see some variations during prime time—when the MCC gets LOX flow thru injectors (before then mainly just have GOX), gives a rapid increase in MCC Pc. Past about 2.5 seconds, should be almost identical to comparisons.
- If any MCFs during START, will simply shut down at that point (on flight, decision is made by shuttle vehicle controller before igniting the solids, and will cause an on-pad abort).
- We used to just ignore MCFs on the test stand, and go ahead and run unless a redline was reached. E.g., during Mainstage, LOX turbine temps redline is 202(?) (both channels must exceed this for three major cycles for a redline). During START, one channel gets a redline for one cycle, it generates an MCF.
- HPFP DS P — See fuel side oscillations. MFV opens at START, liquid hydrogen rushes into the hot nozzle, and flashes into a gas, causing a big pressure increase causing the gas to reach all the way back to the HPFP DS P sensor. Causes a slight oscillation. Benign phenomenon, sometimes happen, sometimes doesn't.
- HPFP DS T — Gradual increase, spikes during fuel side oscillation when you get gas back into the pump. Pump basically stalls momentarily. Pressure goes down. When the fuel pump stalls like this you usually get a warmer start, especially on the LOX side, because the OPB gets less fuel, also giving you a faster start. See some variation due to the efficiency of the pump (vs. the comparison pump). Usually see a spike around 1sec, due to the FPOV 1st notch on the schedule of valve positions during start. The two channels (A and B) are usually plotted separately, because they're located at physically different positions on the preburner (see Fig 3), and the temperature distribution is not perfect.

- FPB Prime — Fill up the LOX dome cavity, start flowing LOX into the injector. Prime is when the face plate is lit all the way across. Measured with HPFP speed; initially running on gas combustion, but at prime get liquid combustion yielding much more energy, causing the speed to jump up. Thus, prime is when the slope of HPFP speed changes.
- Priming sequence is essential, because you want to make sure you start up and shut down fuel rich. So, want to prime FPB first, MCC second, OPB last (need to be about 1/10sec apart). If you primed the MCC first you'd create a lot of back pressure onto the HPFP.
- MCC Prime — when it breaks 100psi. I don't think that's a good way to do it. Rocketdyne defines it at 300psi. It's really just when the slope MCC Pc slope changes.
- OPB Prime — HPFT temp going down initially cause you're running mainly on fuel flow (starting the engine on fuel gas essentially, from fuel flowed through the nozzle), when you get enough LOX flow they turn around and start going back up, and that's the prime time. See a slight increase(?) at the time of the fuel side oscillation when we stall the fuel pump out.
- Mainstage starts around 5 seconds (full closed loop control).
- Pump discharge pressure looks alot like pump speed, so we usually just show discharge pressure at the review.
- HPOTurbine DS T — Slight overshoot; normal. (???)

Shutdown

- HPFT DS T — Curve shape is somewhat a function of how much fuel is flowing through the FPB (?) at the time of shutdown. Fuel flow determines how cold your turbine temps get, and that varies according to FPOV open-loop position (?). Channel A showed a turnaround (see Fig 4), channel B didn't. Probably got a little extra LOX in the FPB dome. Normal.
- Frisbee is the LOX diffuser in preburner dome. Can crack, allowing a little LOX in "there" (?). At shutdown when pressures start dropping, the LOX comes back out and provides a little extra combustion, which will show up in the HPFT DS T as a temp increase at shutdown+5-10seconds at a rate of 6 degrees/sec or more.
- Pump speeds at shutdown. Slowdown a function of resistance; if liquid is still flowing get faster slowdowns than for gas. Shutdown to shutdown+5: Rotor is moving around, got all this pressure on one side of it and all of a sudden you take it off and the impellers tend to push the rotor up. This can bottom the rotor out, which would cause the pump to slow down a lot faster. So, your looking for comparable energy dissipation. If one of your bearings was going bad you might see these come down a lot faster. If speed stops before shutdown+12sec, then usually there's some kind of problem (that's the fastest normal shutdown we've seen).
- HPOTP DS Ts — See a minor turnaround when the purge comes on and forces a little more LOX out, causing MR to go up (see Fig 4). MOV comes closed driving the pressure up, causing a small perturbation (see Fig 4). Mainly want to make sure that this doesn't come down too quickly, indicating that something grabbed in the pump.

Mainstage

- Thrust profile — This test was a little unusual in that we ran 104% without any vents for about 80sec to calibrate the facility flowmeters. Drop down to 65%, 64%, 63% to check for preburner pump bi-stability. Then back up to 104% for awhile, then to 109% and drop to 65% for a 3G throttle simulation. You never want to exceed 3Gs on the vehicle, so during flight you throttle back to 65% as it passes through the point of max acceleration so you don't exceed the 3G limit. Then back to 100% and shutdown. This is a fairly normal acceptance test profile.
- Engine fuel inlet NPSP. NPSP = Net positive suction pressure (inlet pressure minus the vapor pressure; a measure of how likely the pump is to cavitate, the lower NPSP the more likely we are to cavitate). Specs for this value are 6NPSP +2/-0. Just indicates what the facility is doing to the engine. Will see changes in response to changes in engine power level. Plots show a lower bound for this value, which if exceeded would cause the pumps to start cavitating.
- Engine LOX inlet NPSP. Spec is 20 +2/-0. You have to reach this point on a HPOP greenrun to show that you have good margin on the HPOP. Cavitating the LOX pump is a lot worse than cavitating the fuel pump because there is only one impellor, so you would immediately overspeed the pump and lead to a disastrous situation. The HPFP has three impellers, so its a little more tolerant (some people say that the first impellor is always cavitating). The vent is to simulate what happens when the SRBs burn out and the vehicle acceleration drops.

- MR. Fuel flowmeter is not required to be calibrated at 65%, only 100%, 104%, and 109%. So you want to make sure that MR is on the mark at 104%, but it's not as important at 65%. So you see MR vary at 65%. You also see MR spike when you change power levels. You get PC control very quickly after power level changes (fast valve and pump reaction), but it takes MR a few seconds to adjust, usually due to a LOX overshoot.
- FPOV actuator positions. Showed a 2% variation at 104%PL. Think it was due primarily to a different preburner pump, but also due to a different HPFP, and a different HPOTP. The preburner pump efficiency is down on this test, causing its outlet pressure to be lower, causing FPOV and OPOV to be open a little more. FPOV pretty high here, about 88%. Once it gets to about 92-93% you start losing control FPOV, because it's almost all the way open. If you do that on FPOV then you start running off MR, although OPOV will keep PC controlled, so it wouldn't be catastrophic.
- HPFTP DS T — Higher than comparison tests. Probably due to HPFTP being less efficient, which would also account for FPOV being as high as it is.
- Pump efficiency from the data reduction model.
- OPOV. Looks good, so I would expect to see HPOTP DS Temps about the same.
- HPOTP DS T — Looks a little low (about 50deg).

Special Purpose Plots

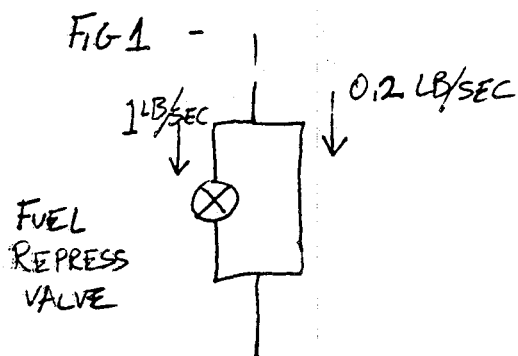
- 2 Sigma Comparisons, at 104% PL when HPFTP DS T is at Max. Look for trends. Mainly looking at ? DS Ts, HPFTP DS Ts and HPOTP DS Ts. HPOTP DS Ts here are low, and HPOP DS P is low (this engine doesn't require as much LOX pressure {more efficient?}, so the preburner pressure and temp are down). As long as everything is within 2 sigma, everything's OK. Here we have HPFTP DS T(?) 2.5sigma high; the fuel pump was pretty bad (efficiency was pretty low), so it's expected.
- Usually see same trends at 109%; same things are happening. FPOV position 2.2sigma high, because of low PBP DS P and low efficiency fuel pump (both contribute to FPOV opening up).
- PBP delta-P vs. MCC Pc plot. Should be pretty constant at each power level. Used to check for bi-stability.
- FPB acceptance test results. Exceeded HPFTP DS T specs by 46 degrees, so that pump was bad. They probably didn't even try to get it accepted.
- Difference between engine and facility fuel flowmeter plot. The engine fuel flowmeter can be bi-stable, in which case you'd see the difference oscillate.
- Heat exchanger. Has an ICD of ? +/-50 at 100% nominal vent. Look at how HEX DS T varies with HPOTP DS T (plot). Project worst-case conditions to 109% to make sure you reach HPOTP DS T redline before the HEX DS T redline. You can orifice the GOX line to control how much LOX goes through the HEX and thus control HEX DS T.
- Low pressure fuel pump DS duct. There is a surge pressure requirement during start.

Observations

- MCC Pc delta problem.

Miscellany

- Hydraulic Lockups. Have done some tests on these recently, where we just pull the hydraulics from the engine. The controller will continue to send commands to the valves, but they don't respond. Once the differentials get to be too large, the controller will switch from channel A to channel B, then to hydraulic lockup. In hydraulic lockup, valves tend to leak which causes them to drift closed. MCC Pc comes down a little (20psi). During a lockup you have to have different ways of analyzing the data, since the controller is no longer taking an active role.
- Bi-stability is that part of the pump curve where you see the pump operating at two different points (see Fig 5), causing oscillations (for the same speed, you get two different delta-Ps).
- Cavitation is when a pump vaporizes the fluid that it's pumping. Can't have any PBP bi-stability.
- Note: The extra delay at 104% to calibrate the facility fuel flowmeter caused an extra complication in comparing this test to the previous one. The best comparison test did not have such a delay, so the analysts had to mentally shift the two curves into alignment (see Fig 6).
- Would be nice to have a program to allow you to cut-and-paste parts of sensor plots from different tests together onto the same plot. This would solve the problem where you have a good comparison test, but it is time-shifted from the current one (as in this test).



FACILITY CONTROL
OF REPRESS, FUEL SIDE
(O₂ SIDE SIMILAR)

FIG 2

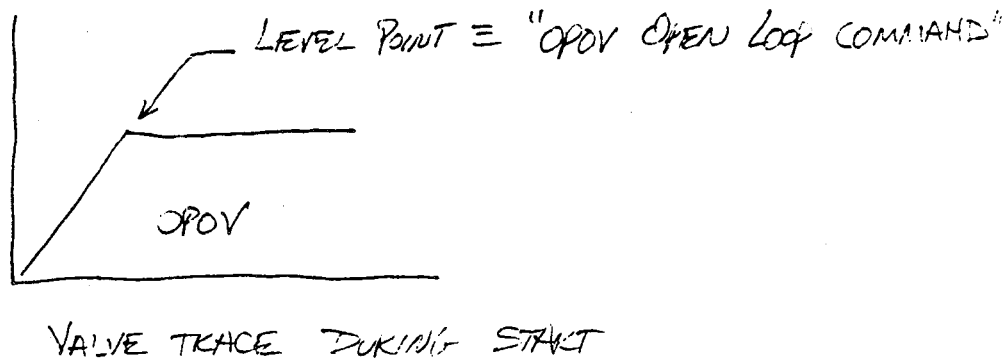


FIG 3

HIGH PRESSURE TURBINE DISCHARGE TEMPERATURE
SENSOR LOCATIONS

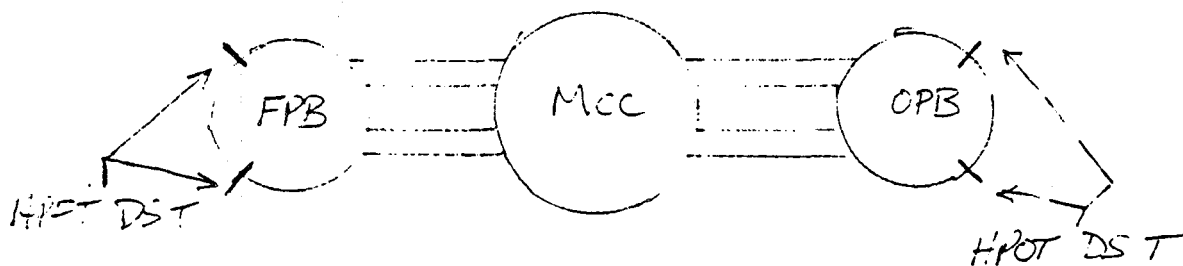
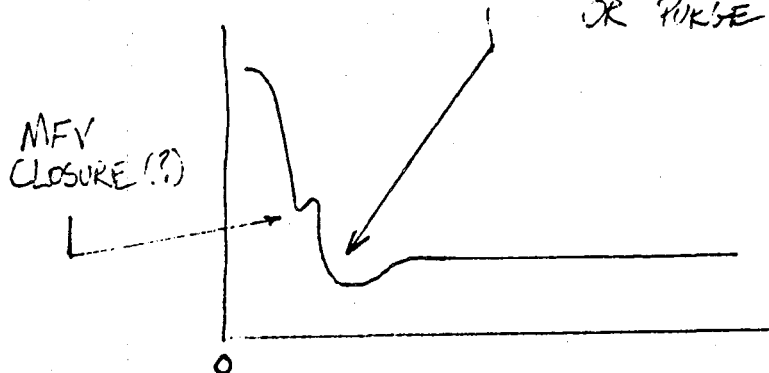


FIG 4

HPDT DST - ② SLOTTED
"TURN-ROUND" DUE TO EXTRA LOX IN DOME
OR PURGE



SIMILAR FOR HPOT DST

FIG 5

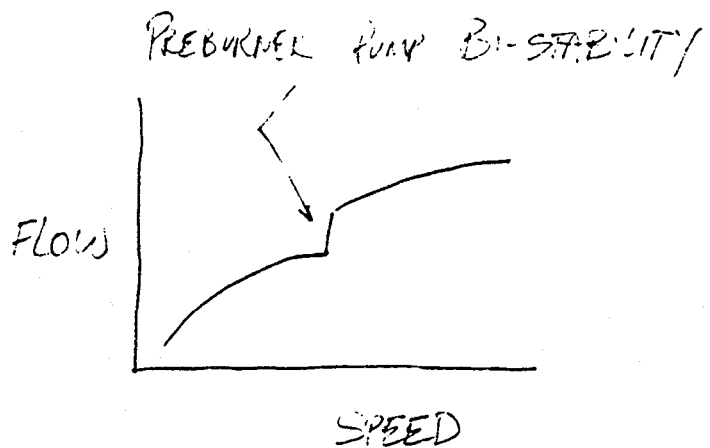
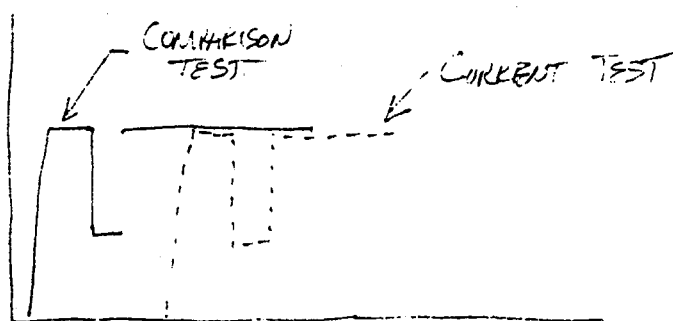


FIG 6

TEST COMPARISON WITH TIME SHIFT



Randy Hurt
4/8/92
Data Analysis

Randy walked me through an analysis of the plots in a complete data package the day after a test (test 904-142, engine 2206). Randy had not seen the data before.

- Randy is responsible for test stands B1 and A2, Taylor is responsible for stand A1.
- Usually Randy, Erik, and Taylor will all look at the data from a test.
- ATD program: alternate turbopump development. Pratt is making an alternate HPOTP (and unofficially making an alternate HPFTP) for the SSME. Scheduled to fly in '96-'97; just now starting to test the HPOTP on the engine. E8 is a facility in Florida where they do standalone testing for the ATD pump. The data is sent to MSFC and Rocketdyne. Major problem with the pump right now is that it has vibration problems. We ran the latest version of the pump last night on B1 at Stennis, and it shutdown early (67 seconds in) due to vibration on heat exchanger.
- Data review is very informal. Randy will start a master observation list.
- The comparison test for this engine had a Pratt HPFTP.
- Randy went through the packages in the following order.
- **NOTE: Unknown to me at the time, my tape recorder died shortly into the interview, so this is very sketchy.**

Quick Look

- Tells if there were any FIDs or special notes. Rest of the information is not important (too imprecise).

Start Comparison Package (1st 6 seconds plotted against comparison test)...

- I like looking at the transients; there's a lot more stuff going on, and you can usually tell more about what's going on in the engine.
- FPOV is high (see Fig 1).
- Fuel side oscillations and HPFT DS T high (see Fig 2).
- HPFT DS T CH B was significantly higher than comparison, but know that CH B on Pratt pumps has typically run low, so this is OK.
- Normal overshoot on HPOT DS T (see Fig 3).
- Normal POGO pressure (see Fig 4).

Start 2 Sigma Package (1st 6 seconds plotted against historical 2 sigma bands)
Shutdown Comparison Package (20 seconds following SHUTDOWN plotted against comparison test)
Shutdown 2 Sigma package
Mainstage Comparison Package
MainStage 2 Sigma Package
Instrumentation (Sets of redundant sensors values plotted together)

Summary

- This entire review takes Randy about 30 minutes to do (of course, there were no major anomalies on this test aside from the vibration, which the systems guys don't seem to worry themselves about).
- Randy's one major observation concerned a significantly higher HPFTP DS T (up 175 degrees on this test relative to the comparison). He conjectured that it was due either to changing the F7 orifice or the change in HPFTPs (both of these noted in the hardware change section). F7 is changed primarily to effect a change on LPFP speed. Randy went up to talk to Brian to see what effect changing this orifice would have on HPFT DS T. Brian looked it up in a table and told Randy that it could account for a 25degree change, max, so Randy dismissed it as a possible cause. To test the pump-change theory, Randy had Brian pull another comparison test, one for the same engine and same fuel pump (it wasn't used as the comparison in the runstream because it was on a different stand and was run a year ago). Randy also checked the

historical database to see if there were ever consecutive tests with such a change in pumps to try and get a ballpark number for how it effected HPFT DS temps.

- Erik: We very often pull additional comparison tests to help diagnose specific problems like this. On the FRF we just had on Endeavor, it had a HPFP DS P was 100psi low relative to the comparison ground test on all 3 engines. We have several theories. One is that on the ground test we have a temperature probe which sticks into the boss where we read the HPFP DS P, but not on flight. So we think that the change in the flow environment around the pressure sensor is responsible for the change. So, we're trying to find a previous pair of firings on which we ran with and without the temperature probe to see if the phenomenon has repeated.

Miscellany

- Erik, discussing another anomaly: If you open up the FPOV, you're going to cut OPOV; that's why you see them interact all the time (???).
- Higher pump inertia will cause a longer start and a slower stop time at shutdown.
- Pratt HPFTP turbine inlets are larger than Rocketdyne's. Thus, changing from a Pratt to a Rocketdyne HPFTP causes OPB PC to go up a little, FPB PC to go way up, and ???.

FIG 1

FPOV IS OPENED UP MORE
(X = THE CURRENT TEST
Δ = COMPARISON)

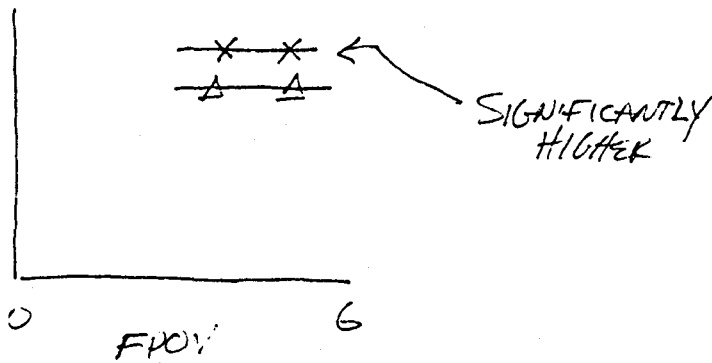


FIG 2

FUEL SIDE OSCILLATION

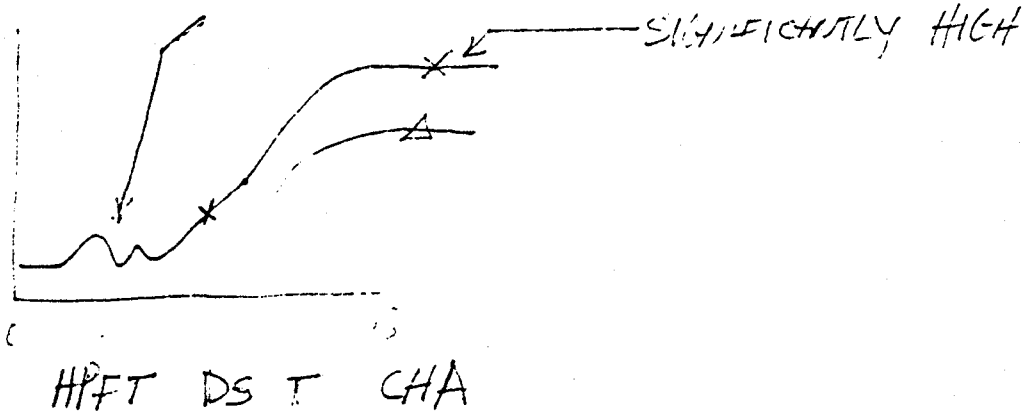


FIG 3

NORMAL OVERSHOOT IN HPFT TEST

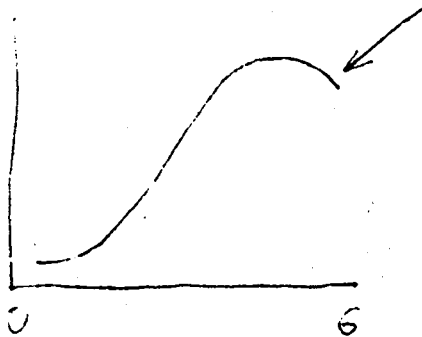


FIG 4

NORMAL YOUNG PRECHARGE PRESSURE

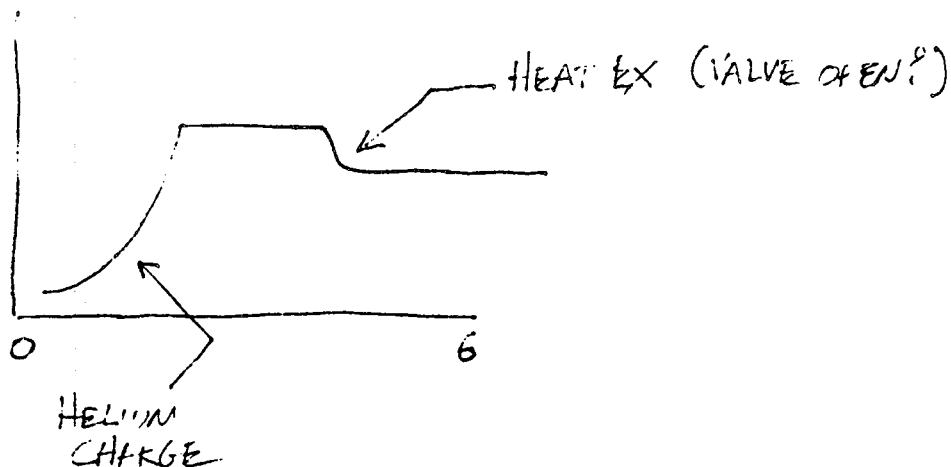


FIG 5

HYFP INLET PRESSURE AVERAGE

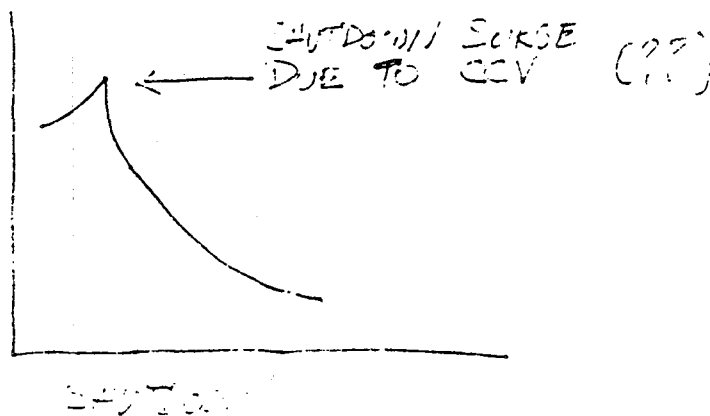
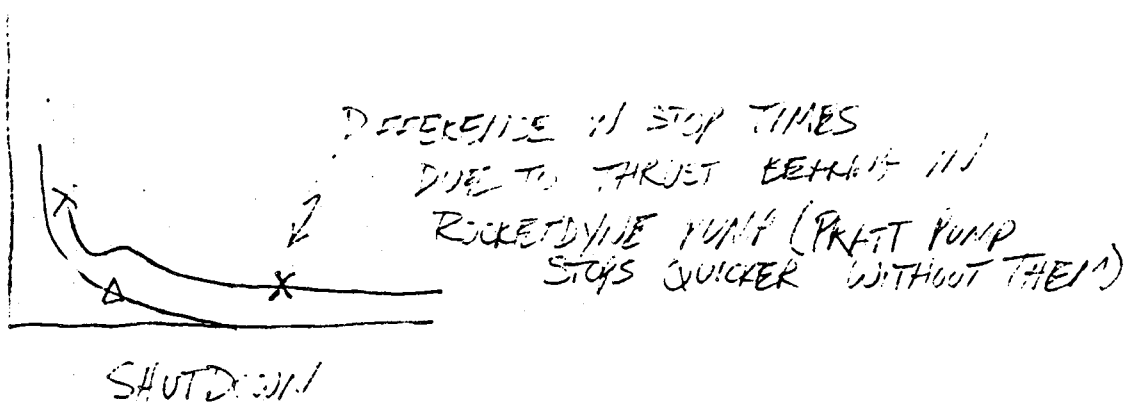


FIG 6

LIFT SPEED



Miscellany 4/8/92

Erik on Analysis of Data from Pre-START and Post-SHUTDOWN

Pre-START

- Want to make sure that you're preparing the engine for a good start. 1. Make sure the cryogenics are in the engine in the way you want them; correct temperature and pressure to start the engine in a consistent manner. 2. Make sure the engine is purged properly. Want to clear out everything including air on the fuel side (air will freeze when liquid hydrogen hits it, and will block lines, etc.). Check for valve leaks: Have skin thermocouples on outside of the engine, and know what the normal instrumentation response is when we drop the cryogenics. So we can tell if there's a leak on the engine or in the engine.
- For flight, we have guys in the HOSC 8 hours before launch. It takes two hours to fill the tanks, after which we basically wait for six hours. Looking for leakage (mainly by skin temperatures downstream of the valves, or internal temperatures downstream of the valves). Check helium pressures, hydraulic pressures, purging.

Post-SHUTDOWN

- Need to make sure we get all of the fuel and LOX out of the engine through purges.
- Check for leaks, especially liftoff seal leakage.
- Check for instrumentation bias. If a sensor does not return to ambient, especially if you have questioned its value, then you can conclude that it shifted off calibration during the test (e.g., a pressure sensor that only returns to 100psi instead of 14psi). Often happens due to thermal effects. Will also check it pre-START to see if the bias existed throughout the firing.

Endeavor FRF Data Review (some Highlights attached)

- Otto Goetz, et al, in attendance.
- Most often asked question relative to anomalies: Have we ever seen this before?
- Can you tell me what effect phenomenon X has on parameter Y?

Interview with Mark Neely (see attached writeup)

- Started as SSME data analyst (test and flight). 7-1/2 years experience.
- Hasn't done regular analysis for the last year.
- Main question: Is the engine ready to be fired again safely?
- If there's a serious anomaly or incident, the everyone gets involved.
- Two major areas of expertise: Pneumatics and Controller (esp. Block I).
- Two major steps: Detection and Resolution. Resolution is hard, if not impossible.

Resolution

- If you see an anomaly during mainstage, check the parameter during prestart and post-shutdown to see if the instrument is biased.
- Look at related parameters. E.g. if a certain pressure is up, then you should look for another parameter which can confirm that the pressure is really up.
- Compare observations with your experience base. Example: There are parameters which always drift, e.g. due to thermal effects.

Resolving "Real" Problems

- Make comparisons with your experience DB. You have to develop a plan to fit the particular anomaly.

Controller

- Runs on 20ms major cycles. Records data 1st 10ms, and then holds data to the end of the major cycle and then posts it. An actual phenomenon will therefore occur some time before it actually shows up in the data.

Pneumatic Actuators

- You probably shouldn't worry too much about this system until you have the "core" diagnostic system developed; it's of secondary concern. Just treat this as an input device to the engine system, but don't worry about diagnosing problems within the PCA.

- Used to shutdown the engine when in hydraulic or electrical lockup (an emergency backup to the hydraulic system).
- Pneumatics also control bleed valves and purges.
- Example anomaly: LN2 dripped onto hydraulic line, freezing a slug of hydraulic fluid, preventing a valve from closing.
- Common problem: Leaking check valves.
- If you have a pneumatic shutdown and have a problem in the PCA it could cause you to not execute the right valve schedules.
- If I were doing an engine system diagnostic module, and I had an increased discharge resistance in one of the big pumps, it would be enough to just say that the problem is a partially closed valve, and not worry about what is causing the valve to be closed.

Interview with Chris Singer

- Possibly the most experienced SSME data analyst at MSFC. However, does not currently look at the data, and is shortly going on a one-year stint at NASA HQ in Washington D.C.
- The most valuable thing the diagnostic system can provide is another set of eyes to look at the data in a consistent manner. That is the primary function of the data analysts here, is to provide an independent look at the data (independent from Rocketdyne). That's also why we have many people look at the data here. It's all to make sure that a problem doesn't go unnoticed.
- Make sure you keep thorough records of all anomalies you see. This would probably solve most of the anomalies you encounter. I still have analysts come to me with "new" problems, when I saw it years ago.

Demo of PC Plotting Package (Jeff Cornelius & Larry Leopard)

- BCCS developed a PC plot program which allows analysts to call up any parameter from any test and plot it on their PC. Several parameters can be plotted together, along with arbitrary algebraic functions of parameter values. You can select a region of the screen with the mouse, and that region will be blown up to the full size of the screen. Anything on the screen can be sent to a laser printer, which puts out better quality plots than the PE. Larry Leopard is currently using the system. Erik saw it for the first time with me and was very impressed.
- See attached network diagram.
- The system is extremely fast, taking only about 15-30 seconds to find the requested data files on the Sun, extract and export the data, and plot it.
- One of the reasons the analysts are reluctant to put the hotfire data into Ingres is that the PC Plot program will not work with it.

6

XXFIENS049 025/G

21

ME-1 MLC LOX DOME TE

△F2ENS049 025/G

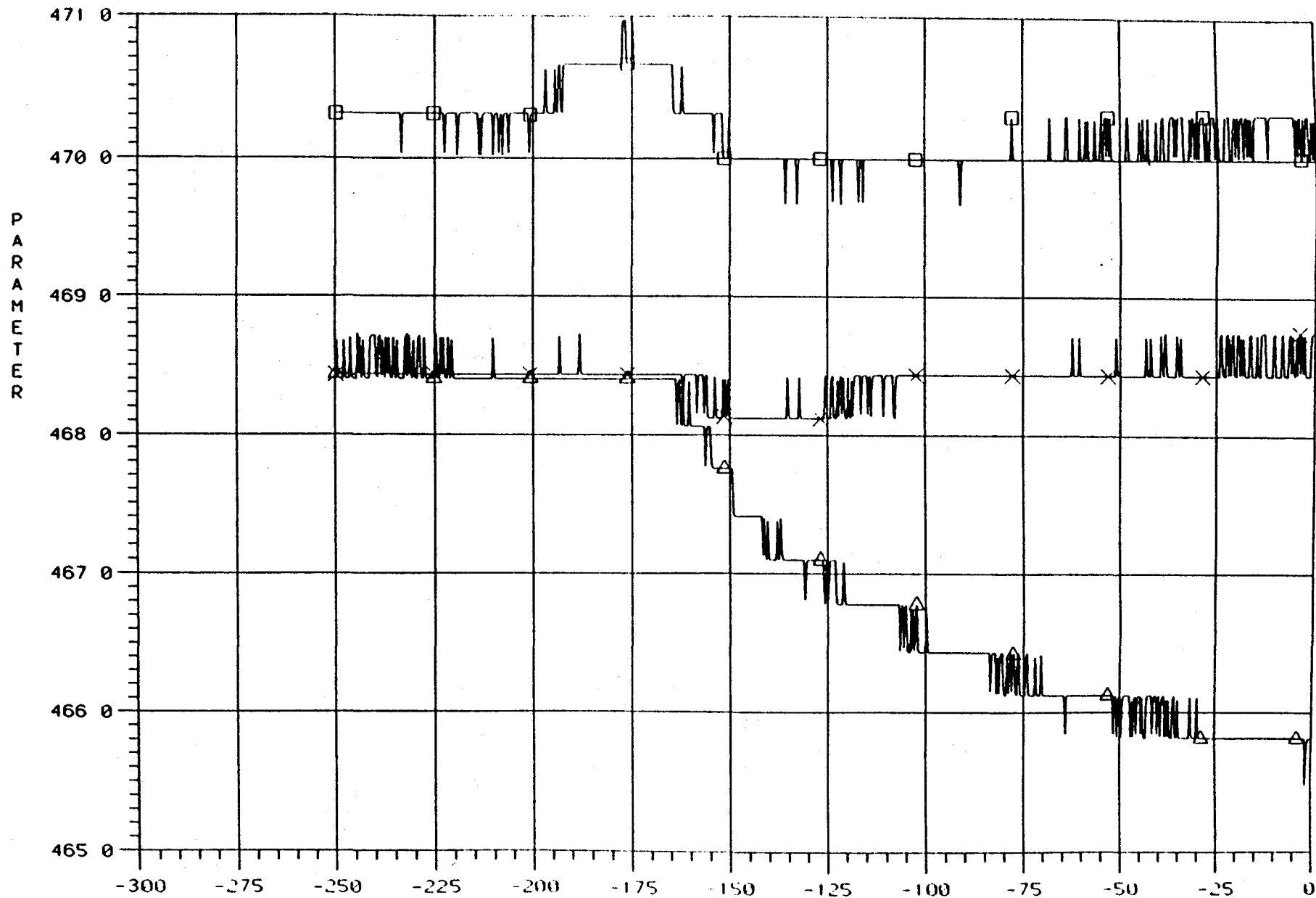
21

ME-2 MLC LOX DOME TE

□F3ENS049 025/G

21

ME-3 MLC LOX DOME TE



POSSIBLE
OX
dome
leak

ENGINE 2035
SHUTDOWN

19 16 SEC

TIME FROM START COMMAND - SECS

VER 4 000

DATE 04 08 '92

TIME 07 27 '95

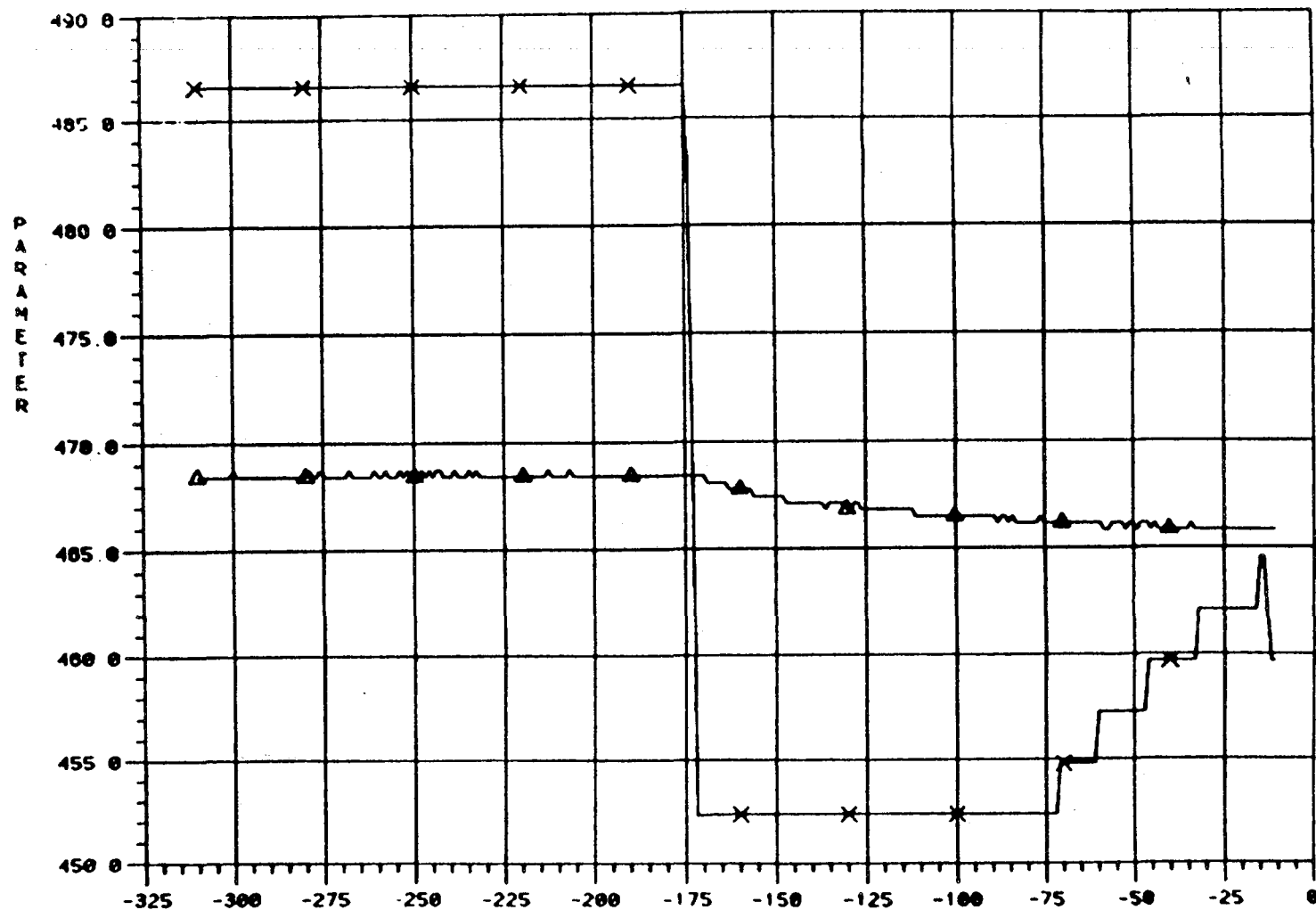


YILC 0 0

GSNT1005A
ME-1T21200

MPENG GN2 PRGR TE MP
ME-2 MCC LOX E TE MP

PLUS 400



} Coincided
w/ Purge

FREN3049 001
ENGINE 0
SHUTDOWN

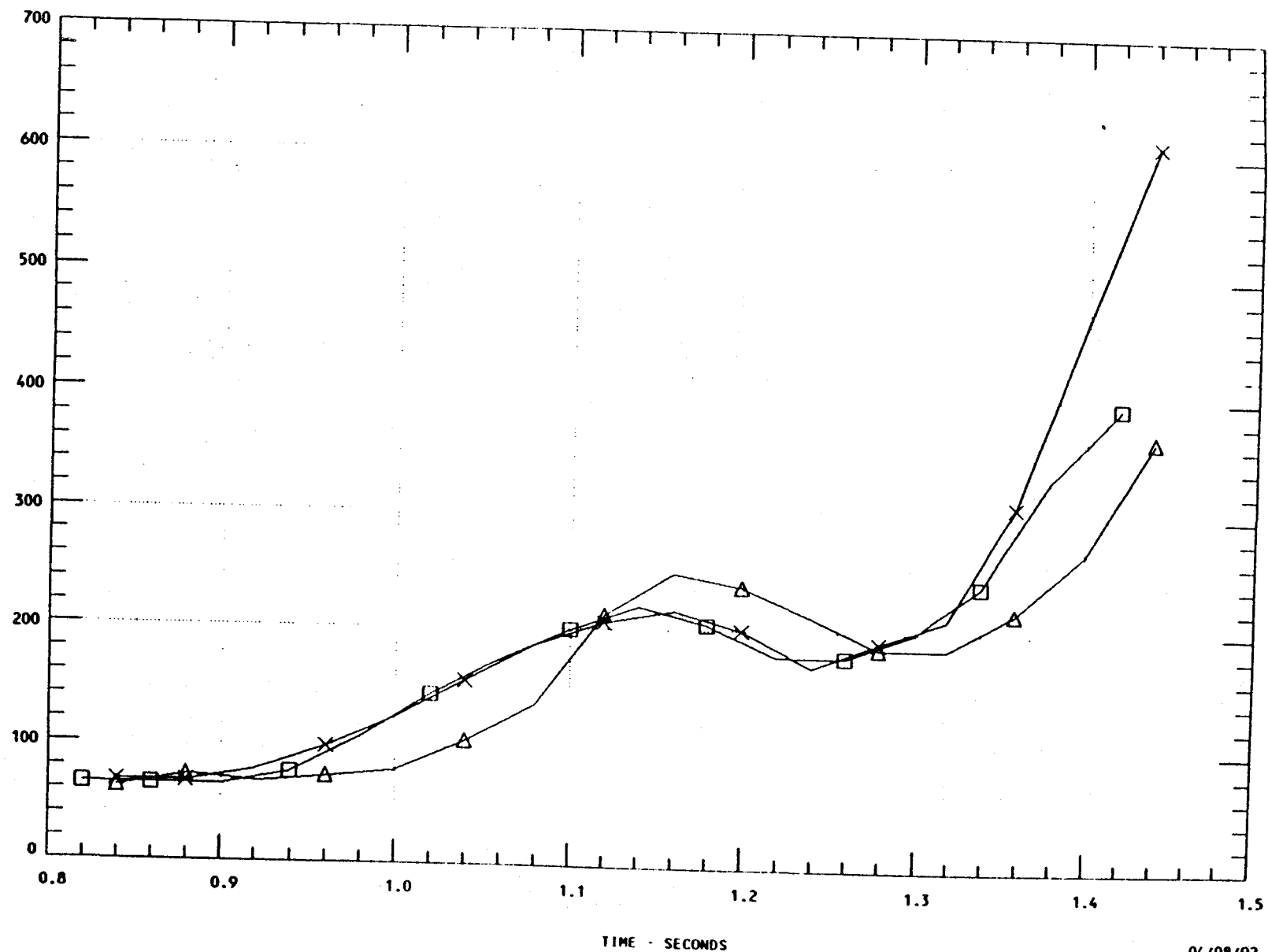
0 00 SEC

TIME FROM START COMMAND - SECS

VER 4 000
DATE 04/07/92
TIME 12 16 32



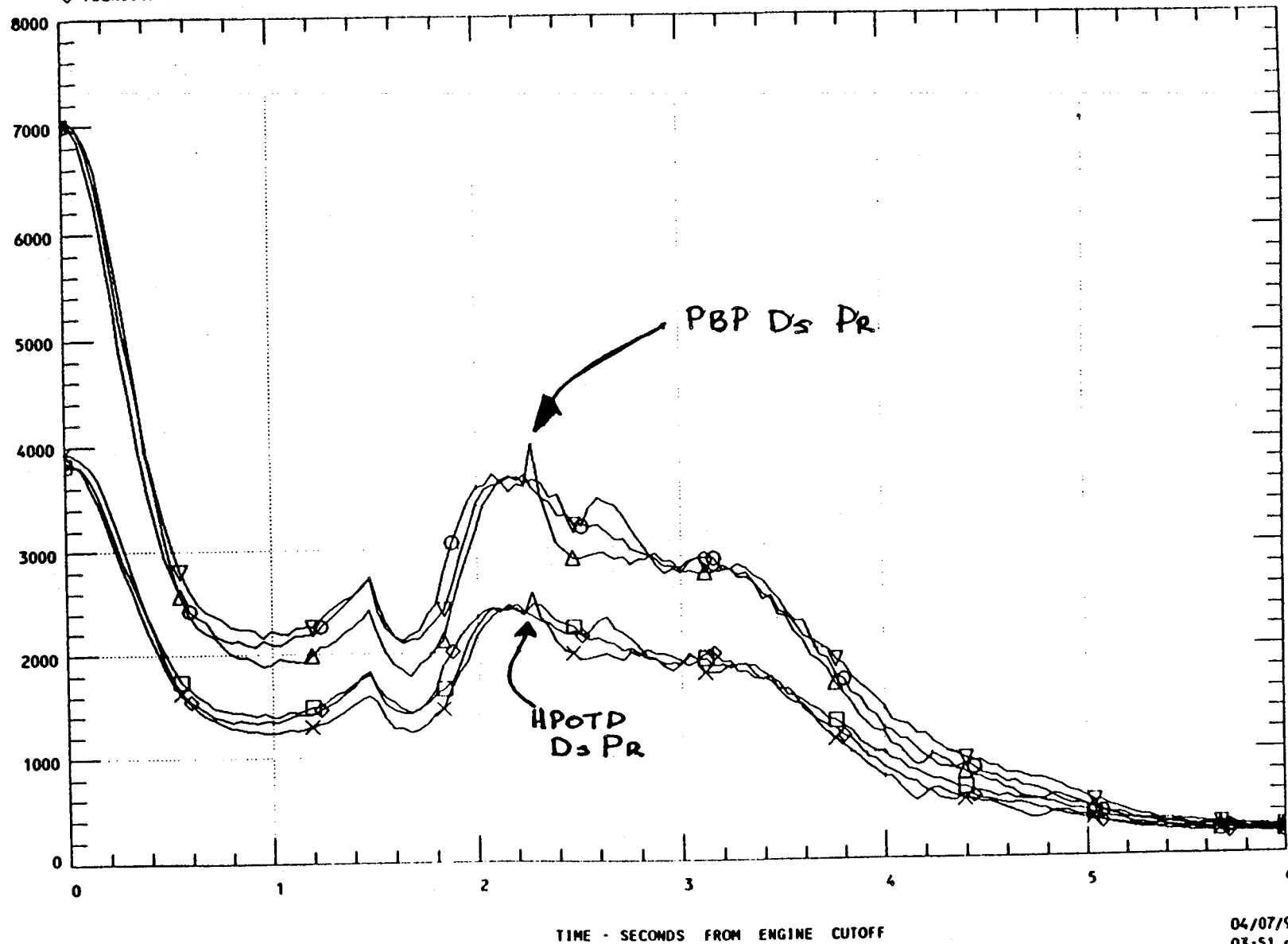
X f1ens049 152 ME-1 HPFP DIS PRESS (PREV) DW 45 PSIA
 Δ f2ens049 152 ME-2 HPFP DIS PRESS (PREV) DW 45 PSIA
 □ f3ens049 152 ME-3 HPFP DIS PRESS (PREV) DW 45 PSIA



04/08/92
 09:03 am
 Fuel side oscillation all 3

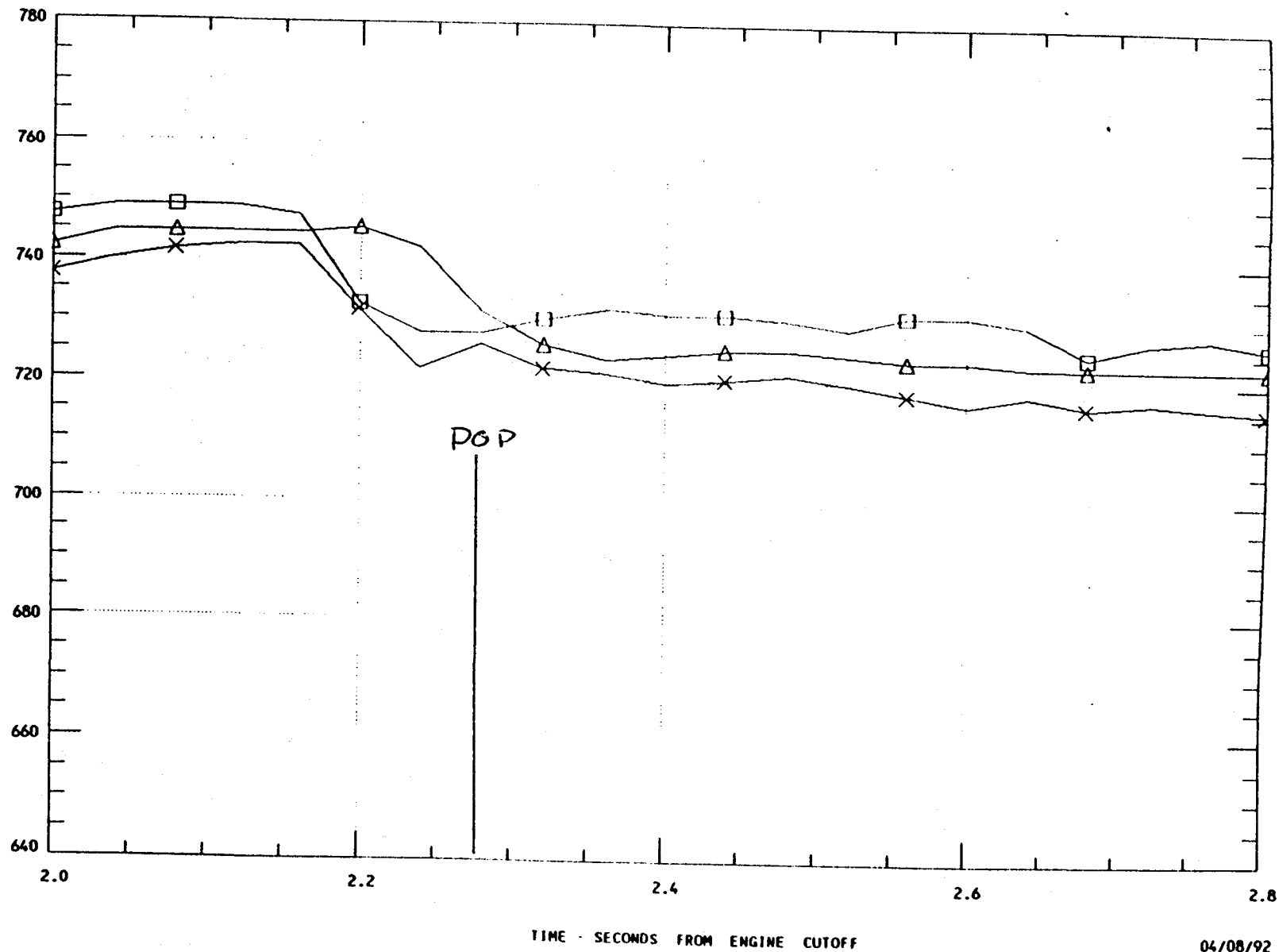
OPB Pol

- | | | | | | |
|---|---------------------------------------|--------------------------|---|---------------------------------------|--------------------------|
| × | F1ENS049 90 ME-1 HPOP DISCHARGE PRESS | DW 30 PSIA (19.16 25.16) | ○ | F3ENS049 59 ME-3 HPOT BST STG DISCH P | DW 33 PSIA (21.92 27.92) |
| △ | F1ENS049 59 ME-1 HPOT BST STG DISCH P | DW 33 PSIA (19.16 25.16) | | | |
| □ | F2ENS049 90 ME-2 HPOP DISCHARGE PRESS | DW 30 PSIA (20.48 26.48) | | | |
| ▽ | F2ENS049 59 ME-2 HPOT BST STG DISCH P | DW 33 PSIA (20.48 26.48) | | | |
| ◇ | F3ENS049 90 ME-3 HPOP DISCHARGE PRESS | DW 30 PSIA (21.92 27.92) | | | |



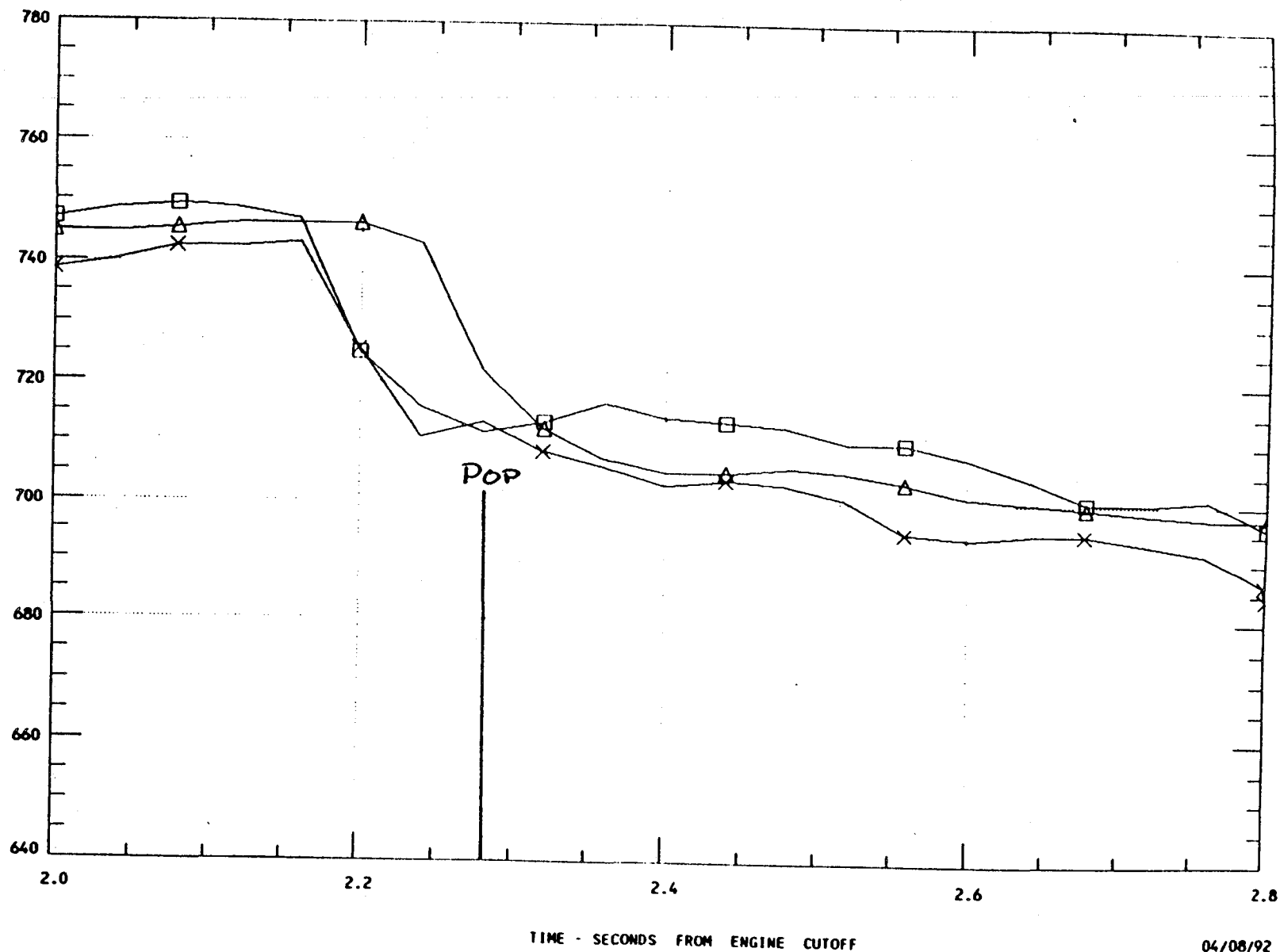
04/07/92
03:51 pm

X F1ENS049 149 ME-1 OXID PREBNR PGE PRESS DW 75 PSIA (19.16 25.16)
 Δ F2ENS049 149 ME-2 OXID PREBNR PGE PRESS DW 75 PSIA (20.48 26.48)
 □ F3ENS049 149 ME-3 OXID PREBNR PGE PRESS DW 75 PSIA (21.92 27.92)



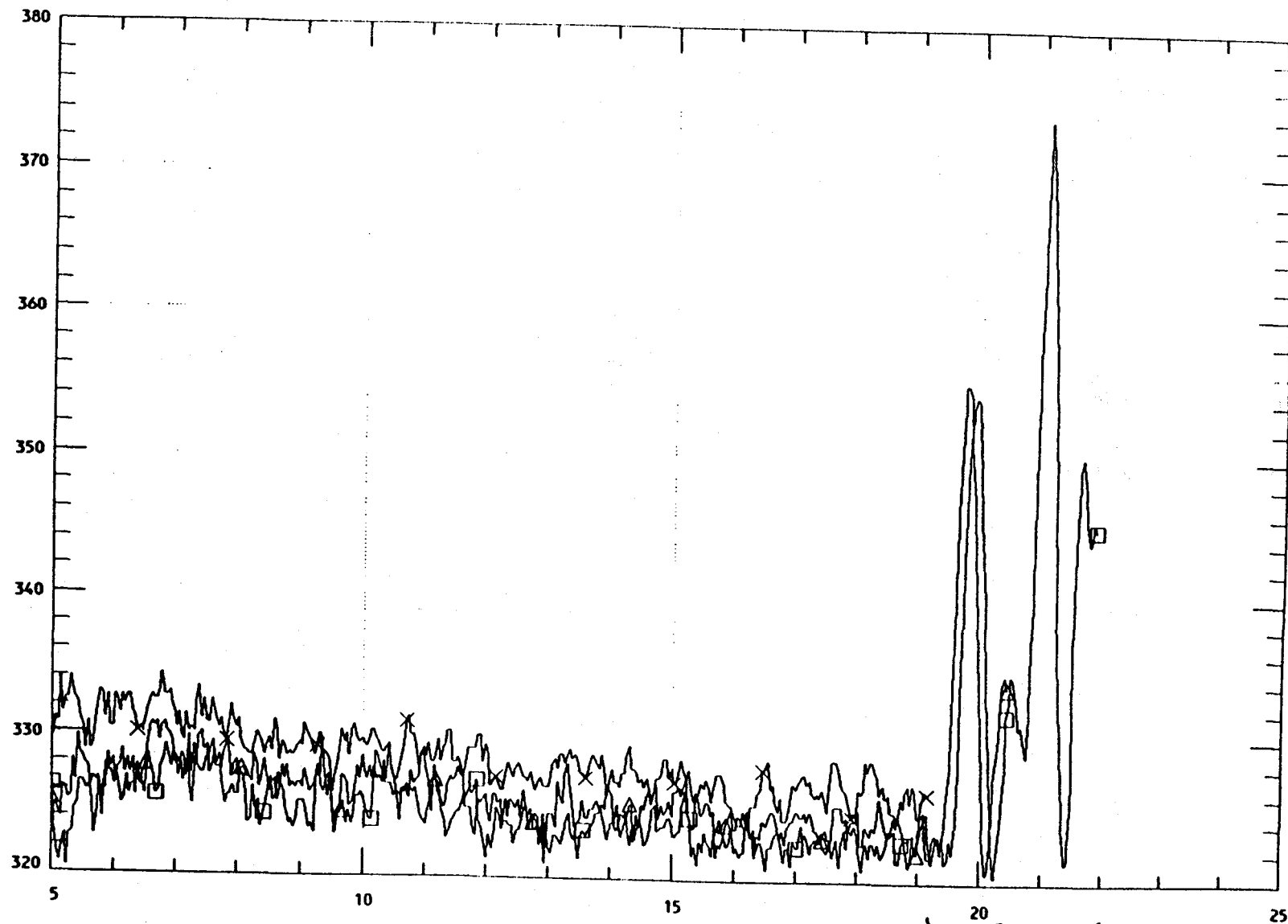
04/08/92
 08:14 am

X F1ENS049 148 ME-1 FUEL PREBURN PGE PRESS DW 74 PSIA (19.16 25.16)
 Δ F2ENS049 148 ME-2 FUEL PREBURN PGE PRESS DW 74 PSIA (20.48 26.48)
 □ F3ENS049 148 ME-3 FUEL PREBURN PGE PRESS DW 74 PSIA (21.92 27.92)



04/08/92
08:12 AM

X F1ENS049 209 ME-1 LPOT DISCH PRESS CH A DW 70 PSIA
△ F2ENS049 209 ME-2 LPOT DISCH PRESS CH A DW 70 PSIA
□ F3ENS049 209 ME-3 LPOT DISCH PRESS CH A DW 70 PSIA



TIME - SECONDS

*Surge due to
Lox Waterhammer
(after El Shuttin)*

04/08/92
09:17 am

Generic Approach to Data Anomaly Detection and Resolution

An anomaly has to be detected before it can be resolved. Some are easy to detect. For example, if an instrument is recording a value outside its range you have an anomaly. More often than not however, detection requires more than just a perusal of the data. Several methods used to detect anomalies are as follows:

- 1) Compare the data with the planned test profile and objectives. Know and understand what the hardware is being commanded to do and check to see that it responds properly.
- 2) Look at the interface conditions. Be able to account for changes in the data as a result of changes at the interfaces.
- 3) Compare the data with an experience base. Use previous tests and historical databases. Make sure to account for changes that might have been made. These include hardware, software, facility and others.
- 4) Check the data against specifications. Look for violations and marginal conditions.
- 5) Look for "unusual" characteristics in the data. These include shifts, drift, spikes and so on.

Anomalies can be classified in general terms as "real" or "instrumentation". "Real" anomalies refers to those related to abnormal performance of hardware or software. "Instrumentation" anomalies are those where hardware or software problems are indicated when there are none. Several ways to determine "real" anomalies from "instrumentation" anomalies are as follows:

- 1) Check the data against the physical limitations of the instrumentation. These include range, response rate, "signature" and others.
- 2) Compare redundant data channels for parameters that have them. It is probably an "instrumentation" anomaly if they do not agree.

~~Some anomalies may be easily resolved. Most cannot. There are several basic steps that should be followed for the resolution of almost every anomaly. These include the following:~~

- 1)

Plot analysis with Eric Sander

The package that you are looking at is a compilation of 8 or 9 packages that are looked at right off the bat (after a ground test). Typical example : Run a test at Stennis (SSC)Space Center during second shift and MSFC receive the data during the middle of the night. Boeing Computer Support Services (BCSS) has a contract to process the data and compile these plot packages. These plot packages are the first look at the data. Plot packages are NOT the first thing looked at for the test but the first look at the data. Typically from one to six people look at the plot packages after a ground test.

The plot analysis package is arranged into four time phases:

1) **Pre-start** - defined as time that data starts being taken until engine start command (T-Zero for a ground test , engine start on flight).T-Zero for flight is SRB Ignition Command Valuable information can be obtained during pre-start that is used for analysis during start, mainstage and shutdown.

2) **Start** - defined as engine start until engine start +5.0 or engine start +6.0 seconds. Engine stabilizes at a given power level in five or six seconds. This means that the main chamber pressure gets up to 3006 psi in five or six seconds. Some parts of the engine don't stabilize for two or three hundred seconds. That means seals or thermal overshoots exist up to that time.

3) **Mainstage** - defined as engine start + 5.0 until engine shutoff command. Shutoff command is normally given by the controller based on a pre-determined time but can be given by redline. Includes throttling, 3-g throttle or bucket.

4) **Shutdown and post-shutdown** - Defined as time from engine shutdown command until up to 300 seconds or so.

Two things preceed the plot package :

1) **Pre-test** - Obtained from Rocketdyne during a telecon. Telecon is held from 3 days until .5 days before the test. First look at the test. The pre-test package is broken into two parts :

A) Rocketdyne's analysis of the previous test including test objectives of the previous test.

B) Objectives , hardware changes and other useful information for the current test.

2) **Quick-look** - After the test is run . A one page fax from SSC with numbers for key parameters to indicate engine operation. Fuel turbine temps , Pc , Lox turbine temps, OPOV position, FPOV position

and discharge pressure on the low pressure fuel pump. Very quick look at how the engine was operating. Contains a rough measure of mixture ratio. Also contains a FID listing if a FID occurred during the test. Recently started getting a phone message with information about the test. FID (Failure Identification Delimeter) is a flag issued by the controller saying that the controller has recognized a problem during the test (pre-start until post-shutdown). FIDS are declared when the parameter has exceeded a pre-set limit that is built into the controller.

Two major catagories of tests are:

1) **Flight tests** - Fleet leader - most amount of starts or time on a particular piece of hardware. Conservative amount of time that can be put on an LRU (Line Replaceable Unit) during a ground test as compared to flight. Abort scenarios are:

A) TAL-Transatlantic Abort Landing. Longest time abort scenario.

B) RTLS - Return To Launch Site

C) ATO - Abort To Orbit

D) AOA - Abort Once Around

2) **Development tests** - All other tested hardware.

These catagories determine objectives and what LRU's are run for a particular test. On flight hardware there is an emphasis on keeping time down on the units. This test, a2-556, is a green-run test. Green run means that a component or engine is run through a ground simulation of a flight environment.

Pre-test for test a2-556 include the Rocketdyne data analysis for test a2-555 (previous test on this stand) as well as the objectives for test a2-556 (Current test on this stand).

Page 1 & 2 (of Eric's viewgraphs) are the MSFC SSME test summary for test a2-556 and include test number, engine number, major components, results for each time phase and observations/anomalies. Reference page 1 & 2.

Page 3 lists the objectives for test a2-556. This is a typical objective sheet for a green-run test. In a development test these objectives would typically be different. This is just used as an example to show the type of information that is derived from a pre-test package Don't get hung up on the words on this sheet. The stated objectives are observed and addressed in the data review. For instance: green run of the pumps will definitely be addressed because the green run

specs call out for looking at the measurements that pertain to the system group to certify whether the pump is acceptable or not. Some of the other stated objectives call for evaluation of certain instrumentation (such as strain gage instrumentation) which systems group cannot make a call because this is outside of our field.

Pre-test gives what we are doing to the engine in terms of the repress flow (fuel side to gox side) and what we are doing in terms of helium and nitrogen pressures (for purges). It specifies what conditions are going to be put on the engine in terms of different fluids.

REPRESS AND FLOWS

In going through the analysis there are actually very few things that we do to control the way that the engine operates. We have very few handles on the engine. There are 5 valves: 1) MFV - Main Fuel Valve, 2) MOV - Main Ox Valve, 3) CCV - Coolant Control Valve, 4) OPOV - Oxidizer Preburner Ox Valve and 5) FPOV - Fuel Preburner Ox Valve. These are used to control the engine. We have what is called the repress control valves. BE CAREFUL USING THE TERM REPRESS. Repress in our field has two connotations. Repress in this case refers to the repressurization flows that come out of the engine (refer to the attached engine schematic) In flight gaseous hydrogen goes to the tank and is used to pressurize the ullage in the tank and force liquid down into the engine. Certain conditions have to be met at the low pressure pump inlet (LPFP). One of these conditions is that a certain amount of pressure has to be maintained at the pump inlet to keep from cavitating the pump. Cavitating the pump means that certain gaseous zones are forming going into the pump. This can very quickly ruin the pump or destroy the engine. Pressure going into the low pressure pumps is maintained by the ullage pressure. Ullage pressure is the gaseous pressure above the liquid in the tank. This is the fuel repress flow.

The GOX (gaseous oxygen) repress flow is on the other side. The liquid hydrogen comes in through the LPFP down to the HPFP and gasifies right downstream of the MFV due to the temperature and pressure conditions that exist in the engine. The LOX (Liquid Oxygen) is liquid throughout the entire system until it reaches the preburners. Liquid can't be pumped back into the tank to maintain ullage. It has to be gas. What happens is that the LOX is run through

a heat exchanger (HEX). A heat exchanger is simply coils that are contained within the preburner. The LOX enters the HEX and is gasified and this GOX is used to pressurize the LOX tank. On a ground test this GOX is just dumped overboard. The goal is to simulate the parasitics that exist, to the tank, during flight. The fuel, during a ground test, is burned in a burn stack.

In flight we have to control the flow because you can't just dump an endless amount of flow into the ullage. The tank will be overpressurized. What we have is a flow control valve that gives either max repress flow or min repress flow depending on where the valve is. We also have the ability to give no repress flow. We also have the ability to give nominal repress flow. Nominal is kind of a misnomer because in flight there is either max or min repress flow. Normally, on a ground test, a test is either run at max or min repress flow. Repress valves are shuttled back and forth to give max or min repress flow. 1.2 lbs/second is considered max repress flow on the fuel side. Downstream in that line there are orifices that give 1.2/lbs at RPL (Rated Power Level). Min repress flow on the fuel side is .2 lbs/second. On the LOX side max repress is 2.3 lbs/second . Min repress is 1.1 lbs/second.

There are things done to the inlet pressures called pressurization. This will be shown during the mainstage portion of the package. Repress is also referred to when inlet pressures are increased. The cryogenic conditions to the low pressure pumps inlets are another area that can be controlled(temps and pressures). These are the portions that we have a handle on during a test.

In doing analysis especially during the hotfire time, start until shutdown plus 10 - 20 seconds, we know what is being done to the engine, what the effects are and what to expect from the data. This is the main information used to detect an anomaly. It has to do with the gains. We know what the gains are if repress flow is varied. Anything seen outside of the realm of these gains are considered an anomaly or are initially recognized as an anomaly.

Helium is used primarily to purge the HPOP intermediate seal and to shut the engine down with pneumatics if the hydraulics have a problem.

Nitrogen is not used during a flight. It is used pre-start to purge the LOX side of the engine. Hydraulics are used to run the valves.

Every once in a while there is an interesting piece of information from the pre-test. Example: recently we have been dripping liquid air on the hydraulic lines or problems exhibited on previous test are shown. This is very important information that may be hidden among the test procedures in the pre-test. This is important because this has to be looked for in the data.

One of the most important pages in the pre-test is the hardware change page. This is important because hardware changes may change the way that the engine operates. If the hardware change is known it can help determine what is or isn't an anomaly. The thought process is that anything that is different from a previous test or previous experience is an anomaly in our minds. This allows the disqualification of "mind anomalies". Each piece of hardware has its own characteristics and will change the way the engine operates.

In this test 3 of the 4 pumps were changed, controller, fuel duct. Some of the hardware changes are more important in terms of engine operation than others. For example: Negligible engine operation change is expected from changing the HPV (Helium Pre-charge Valve). On the other hand if a pump (HPOP, HPFP,LPFP, LPOP) is changed there will be good amount of engine operation change. Turbine temps will change, pressure's will change because of the change of pumps. No two pumps are the same.

There are two controllers :

- 1) Block I
- 2) Block II

There will be negligible effect on engine operation due to the change out of the controller.

HPFD (High Pressure Fuel Duct) now made of two different materials

- 1) Titanium
- 2) Inconel

The two ducts have a different inside diameter. The Inconel duct is bigger by a little bit. Static pressure in this line is 100 psi lower with an Inco duct. This will cause fuel pump discharge pressure to change by 100 psi.

End of Pre-test information...

①

SSME TEST SUMMARY

TEST NO: A2-556

ENGINE NO: 2107

CONDUCTED: 4/22/92

PROGRAMMED DURATION: 300.00 sec
C/O: Programmed

MAJOR COMPONENTS

HPFTP: 4209
LPFTP: 9305
HPOTP: 4012
LPOTP: 2128
PWRHD: 2014
MCC: 4013
NOZZLE: 2016
CONTR: F48

RESULTS

Prestart - Satisfactory
Start - Satisfactory
Mainstage - Satisfactory
Shutdown - Satisfactory

A3-51

A2-556 OBSERVATIONS/ANOMALIES

ENGINE SYSTEMS

START - Nominal

MAINSTAGE - M/R ~ 6.03 @ 104%

- HPOTP(U/N 4012), LPOTP (U/N 2128) Passed preliminary greenrun requirements.
- HPFTP(U/N 4209) shows indications of rotor hang-up. Turbomachinery to RID.

SHUTDOWN - Nominal

INSTRUMENTATION

Facility HPFP Speed (pid 764) - Not reading throughout test. CADS speed good.

MCC Liner Cavity P #3 (pid 1957) - Erratic nature throughout mainstage may indicate sense line leakage.

(3)

APR 31 '92 7:06

TEST 902-556

• OBJECTIVES - ENGINE 2107

• ENGINE 2107 HPOP GREEN-RUN TEST

• 300 SECOND TOTAL DURATION

- 52.0 SECONDS @ 109% RPL
- 109.8 SECONDS @ 104% RPL
- GREEN-RUN OF HPOP U/N 4012, HPFP U/N 4209, & LPOP U/N 2128
- GREEN-RUN OF CONTROLLER U/N F48, AFV LINE, NOV, AND HPV
- GREEN-RUN OF 5 ACTUATORS (5th FLIGHT SET - ECP 1202)
- CERTIFICATION EXTENSION OF AAAA13 SOFTWARE
- EVALUATION OF HPOP WELD 3 INSTRUMENTATION
- EVALUATION OF MAIN INJECTOR LOX INLET TEE INSTRUMENTATION
- EVALUATION OF OPB LOX DOME INSTRUMENTATION

φ

PAGE 004

10A

- FUEL REPRESS FLOW CONTROL VALVE
 - OPEN PRIOR TO E/S
 - 1.2 LB/SEC @ RPL (MAX)
- GOX REPRESS FLOW CONTROL VALVE
 - OPEN PRIOR TO E/S
 - 2.35 LB/SEC @ RPL (MAX)
- ENGINE He INTERFACE PR
 - 740 $\pm$ 10 PSIA
- ENGINE GN2 INTERFACE PR
 - 600 +50 /-20 PSIA
- HYDRAULIC INTERFACE PR
 - SUPPLY PR DURING CHILL = 475 $\pm$ 25 PSIG
 - SUPPLY PR PRIOR TO PSN 4 = 3085 $\pm$ 50 PSIG
 - RETURN PR = 100 $\pm$ 20 PSIG @ 2 GPM

- STANDARD TEST PROCEDURES

- FACILITY GN2 HEATER SCHEDULE
 - SAME AS LAST TEST (REF TEST 902-553)
- OXIDIZER CHILL PROCEDURE
 - SAME AS LAST TEST (REF TEST 902-553)
- CHILL INSPECTION REQUIREMENTS
 - VISUALLY INSPECT VALVE ACTUATOR AND ACTUATOR HYDRAULIC RETURN LINES FOR LIQUID AIR IMPINGMENT AT LEAST 30 MINUTES AFTER FUEL DROP. ANY HYDRAULIC COMPONENTS THAT ARE SUSCEPTIBLE TO LIQUID AIR IMPINGMENT WILL BE INSULATED OR SHIELDED
- MFV HEATER SCHEDULE
 - SAME AS LAST TEST (REF TEST 902-553)
- ENGINE GN2 RING PURGE SCHEDULE
 - SAME AS LAST TEST (REF TEST 902-553)
- CONTROLLER COOLANT PURGE, MINIMUM OF 10 SCFM GN2
 - SAME AS LAST TEST (REF TEST 902-553)
- PRETEST AVIONICS CHECKOUT PROCEDURE
 - SAME AS LAST TEST (REF TEST 902-555)
- GLOW PLUG OPERATION
 - SAME AS LAST TEST (REF TEST 902-553)
- POST SHUTDOWN LH2 LOCKUP PROCEDURE
 - SAME AS LAST TEST (REF TEST 902-553)

A3-55

TEST 902-556

• HARDWARE CHANGES

- REPLACED HPOP (ES-9839)
 - REMOVED U/N 2218, P/N 8R032874-11, S/N 4875860
 - DEVELOPMENT UNIT
 - INSTALLED U/N 4012, P/N RS007701-761, S/N 2185620
 - GREEN-RUN UNIT
- REPLACED HPFP (ES-9840)
 - REMOVED U/N 2033, P/N RS007501-1151, S/N 4862150
 - COMPLETED GREEN-RUN
 - INSTALLED U/N 4209, P/N RS007501-1141, S/N 3041965
 - GREEN-RUN UNIT
- REPLACED LPOP (ES-9839)
 - REMOVED U/N 4203, P/N G1RS007801-191, S/N 4874673
 - DEVELOPMENT UNIT
 - INSTALLED U/N 2128, P/N RS007801-191, S/N 4866312
 - GREEN-RUN UNIT
- REPLACED CONTROLLER (ES-9841)
 - REMOVED U/N F52, P/N RE1493-12, S/N 222
 - COMPLETED GREEN-RUN
 - INSTALLED U/N F48, P/N RE1493-12, S/N 218
 - GREEN-RUN UNIT
- REPLACED HPFD (ES-9840)
 - REMOVED P/N R035533-1, S/N 2079550
 - COMPLETED GREEN-RUN
 - INSTALLED P/N R035533-1, S/N 2079541
 - DEVELOPMENT UNIT
- REPLACED HPV (ES-9841)
 - REMOVED P/N RS010180-291, S/N 4876759
 - COMPLETED GREEN-RUN
 - INSTALLED P/N RS010180-301, S/N 4877206
 - GREEN-RUN UNIT

6

APR 21 '92 7:06

PAGE.005
P.05

14R1

(7)

TEST 902-556

04-22-1992 07:37

618 710 5181

TEST OP'S

19

P.17

• HARDWARE CHANGES CONTINUED

- REPLACED AFV LINE (ES-9841)
 - REMOVED P/N RS007083-261, S/N 4874593
 - PREVIOUS GREEN-RUN UNIT TO SUPPORT TEST
 - INSTALLED P/N RS007083-261, S/N 4875731
 - GREEN-RUN UNIT
- REPLACED ALL 5 ACTUATORS & MOV (ES-9844R1)
 - INSTALLED 5th FLIGHT SET (ECP-1202) ACTUATORS FOR GREEN-RUN
 - REMOVED MFVA P/N RES1008-8122, S/N 118
 - REMOVED MOVA P/N RES1008-5119, S/N 095
 - REMOVED CCVA P/N RES1008-7319, S/N 054
 - REMOVED OPOVA P/N RES1008-6116, S/N 114
 - REMOVED FPOVA P/N RES1008-6116, S/N 121
 - REMOVED MOV P/N RS008255-401, S/N 4872458
 - COMPLETED GREEN-RUN
 - INSTALLED MFVA P/N RES1008-8122, S/N 088
 - INSTALLED MOVA P/N RES1008-5119, S/N 055
 - INSTALLED CCVA P/N RES1008-7319, S/N 033
 - INSTALLED OPOVA P/N RES1008-6116, S/N 066
 - INSTALLED FPOVA P/N RES1008-6116, S/N 092
 - INSTALLED MOV P/N RS008255-401, S/N 4881816
 - SWAPPED IGNITORS (ES-9850)
 - REMOVED OPB #1 IGNITOR P/N 61R0013000-1, S/N 2107714
 - FAILED TO QUENCH
 - REMOVED FPB #2 IGNITOR P/N R0014020-021, S/N 4924788
 - INSTALLED OPB #1 IGNITOR P/N R0014020-021, S/N 4924788
 - INSTALLED FPB #2 IGNITOR P/N 61R0013000-1, S/N 2107714
 - HIGHER FPB PR WILL HELP QUENCH IGNITOR
 - MAIN INJECTOR LOX POST BIASING (ES-9820)
 - CHANGED BIAS REQUIREMENT TO 0.107-.111 (WAS .100-.104)
 - 29 POSTS REQUIRED ADDITIONAL BIASING TO MEET REQUIREMENT

TEST 902-556

φ

- SOFTWARE CHANGES - SAN DATED 4-21-92 REV 1, FEC 10076
 - REVISED SCALING COEFFICIENTS FOR NEW CONTROLLER AND ACTUATORS
 - INCORPORATED ACCEPTANCE TEST MPOT DS TEMP REDLINES
 - GREEN-RUN HPOP
 - L/C REDLINE (2.3-5.8) = 1660°R
 - M/S RESLINE (5.8-C/0) = 1660°R
- INSTRUMENTATION CHANGES
 - DELETED THE 2 ADDITIONAL LPFT INLET PR MEASUREMENTS (ES-9867)
 - REMOVED MCC COOLANT DISCHARGE PRESSURE REDLINE
 - DELETED 4 HPOP ISOLATOR STRAIN GAGES (ES-9839)
 - INSTRUMENTED PUMP REMOVED
- FACILITY CHANGES
 - REVISED FACILITY SYSTEMS TO SEND SIGNAL THROUGH FACILITY CUTOFF RELAY (ES-9856) PRIOR TO THE CADS ADVANCE AND FACILITY READY RELAYS
 - SYSTEM CHECKOUT COMPLETE AND SATISFACTORY
 - PREVENT POSSIBLE TIMING PROBLEM
- FACILITY AND FACILITY MONITORED ENGINE REDLINE CHANGES
 - DEACTIVATED LPFP RAD ACCEL AND HPFP SPEED REDLINES
 - COMPLETED 1ST TEST OF NEW LPFP BUILD
 - DELETED SPECIAL MCC COOLANT DISCHARGE PRESSURE REDLINE (ES-9867)
 - COMPLETED REQUIRED 3 TEST WITH REDLINE (NFL 2563)
 - ACTIVATED LPOP RAD ACCEL REDLINE
 - INITIAL TEST OF NEW BUILD

16R2

9

TEST 902-556

04-22-1992 07:37

818 710 5181

TEST OP'S

P.18

• KEY PARAMETERS

- ALL REDLINE MEASUREMENTS
- MPFP GREEN-RUN MEASUREMENTS
 - SAME AS LAST TEST (REF 902-554)
- HPOP AND LPOP GREEN-RUN MEASUREMENTS
 - SEE FOLLOWING CHARTS

• FLOWCHECK RESULTS

- OPOV D/S SLEEVE FLOW CHECK, OPOV S/N 4892654 / OPOVA S/N 066
 - COMP LEVEL = 24.2 SCFM
 - ENG LEVEL = 22.9 SCFM
- FPOV D/S SLEEVE FLOW CHECK, FPOV S/N 4871490 / FPOVA S/N 092
 - COMP LEVEL = 28.5 SCFM
 - ENG LEVEL = 32.5 SCFM

4

Tape 2 Erik Sander Interview - Plot Analysis

So we take the facility flowmeter, take the Kf to match the two flowmeters. C2 is another way of saying fuel flow. Most people think that if you vary C2 you are setting the mixture ratio on the engine. You are not doing that. If you set a C2, you set a given fuel flow. I have plotted curves that say that if you use this C2 at a given power level, you will get this fuel flow. PERIOD. Unless you are in hydraulic lockup, you will get it. If the controller is controlling the engine, you will get that fuel flow. To get mixture ratio, we set the fuel flow so that with 6 times that amount of lox flow, we will get the desired chamber pressure. This goes into how the engine operates and how we understand the system. I guess that this is as good a time as any to go into this. You must have this basic understanding of how the controller operates and how it affects the engine system, to get a basic understanding of the engine itself. Remember I told you that there are 5 valves? In mainstage, there are only 2 valves that are used to control the engine, and they are the two preburner valves. If you can understand the drivers of those two valves, you have a very good understanding of how the engine system is going to respond to the controller.

I told you that C2 sets fuel flow. It does that through modulating the FPOV. The FPOV provides lox to the HPFT, and therefore powers the pump end up or down to given a certain fuel flow. It goes and reads the volumetric flow rate, temps and pressures, converts to a mass flowrate, multiplies that by the C2, compares the actual to what is desired, is this the flowrate that I want to be running at? If not, it opens the valve more, speeds the pump up, you get more mass flowrate through, and it modulates to bring the two together. Not all of the fuel goes to the chamber. You lose some through leaks in the seals, and of course you lose some through repress. But the majority does end up in the chamber to be burned.

The controller has two hands, one doing the fuel flow thing, the other controls the OPOV, the other preburner valve. The only thing its looking at for that is the MCC Pc. It looks at the Pc and the commanded Pc and it says "Match those two." We build a thrust profile into the controller, before the test, that says at these times we will change to these pressures, MCC Pc's. So the controller knows what power level it is supposed to be at, Pc ref, and it may be low, say 3000 psi, so it's going to open the OPOV to provide more lox flow to the preburner which provides more energy to the HPOT and HPOP, runs more lox flow, makes the Pc come up. If you know the efficiency of the chamber and you set the fuel flow right and the lox flow right, you end up getting two things: you get Pc at the point it's supposed to be at and you get your mixture ratio right, i.e., 6# lox for every 1# fuel. C2 sets the fuel flow, but it's an indicator of chamber efficiency. The more fuel you have to run, the lower the chamber efficiency and the more propellant you have to run. That is usually in the software changes.

Claudia: There are some pages in the pre-test which contain a box with this information?

Erik: Yes. Each pretest contains the C2 and Kf that will be run on that test. C2 varies with power level, because the chamber has different efficiencies with different power levels. Kf varies. Sometimes we have a flat flow, sometimes we don't. It really depends on the efficiencies of that turbine. By flat flow I mean that when you go up in power, you may have a different efficiency in the LPFT, so you'll get a different Kf.

Tim: And you also said that C2 changes with power level?

Erik: Yes. For that they have a slope that has been built in forever. There is an equation in the controller, it's a straight line. And for different engines you move the line up or down. We never change the slope of the line though. In fact it's not really a straight line, it's a curve. But for our range of interest it approaches a straight line. All of that info is in the pretest. Our models group makes recommendations for C2 and Kf for every test. Rocketdyne does the same, and we compare to be sure that we are not too far off.

Instrumentation changes are important because they tell us what extra things that we need to be looking at. In systems, you won't be too interested in a lot of this because it isn't stuff that we normally use to analyze the system. We may put extra instrumentation in there to analyze something on this test, but it's not something that you can build into a module because it's not going to be there for every test, by any means. Most of this is extra stuff, like here we added a couple of turbine inlet pressures. Facility changes may affect the engine to some degree, but not too much, for example the facility flowmeter. It will not affect the way the engine operates, but it affects the way we see the data. If it's a new flowmeter that has been calibrated at a different calibration constant, it will tell us that the engine is running slower than the engine is actually running. We'll go off crazy trying to analyze, and then we'll realize they changed the flowmeter. Redline changes ... You have to keep in mind that the only way we can tell what the engine is doing is by looking at the data, and if the data is bad then we are in trouble. Key parameters are nothing to worry about, they're things they have to have in order to run the test.

There are a lot of other packages in the pretest that I'm not going to go over. If you decide later in the week that you want to go over them, let me know.

Let's start with Pre-start. We break them out into different packages because there are 4 of us sitting there, and we can let different people look at different packages at the same time. It's a lot quicker that way. This is a package of data that we go through. As I go through I'm going to try to point out different things that we look for as we are going through a test. What

drives the changes in the parameters. If there is something you are interested in, please stop me and I'll try to explain it. On some of these plots, you only have 1 test, on others you will see 2 or 3 plotted. Depends on whether you want to compare to another test or tests. For example on this first plot, you have two different pids from the same test. Be careful to know what you are looking at. Here they give you the symbol, the pid number, a brief description, and the ranges. The T indicates it's reading a temperature, I'm not sure what the 2 means, I think it has something to do with what they stick it on with. On the bottom, you get the test number, the engine number, shutdown time, and some other extraneous data. That's the general format. If there had been multiple tests, the labels at top would have shown them instead of the multiple pids listed here.

We will use this schematic to find out where things are in the engine and to relate different perturbations in the data.

June: (She asked something about the range of the instrumentation)

Erik: We hope that the instrumentation range is much broader than the engine range. If we get outside the range of the instrumentation, you don't want to believe anything that you see. We very rarely go outside of those ranges, if we do it's because ????. The MFV DS temp is just down stream of the MFV. In prestart, we break things up into 4 purge sequences. We have appropriate documentation for each of these phases. PSN1, PSN2, PSN3, PSN4, engine ready, start enable and start. These phases are the sequences we go through to prepare the engine to receive propellants and also to go ahead and start reliably. Reliably means repeatably and safely. During the different purges, we purge different parts of the engine with either He or N2 in different ways. Most of that is documented in our book. Cryogenics are flowing into the engine prior to start, through the LPFP, separates at the turbine, comes through the HPFP and comes down and sits right at the MFV. So in pre-start that is where the fuel is. This fuel bleed valve allows a small amount of fuel to run out and go to the burn stack. It's function is to keep fuel running through the system. You can't allow these cryogenics to sit static in one place because they gasify, both the lox and the fuel. The lox prestart comes down through the LPOP, comes through the duct. Some goes to the pogo, other feeds through the HPOP, through to the AFV that is driven open by pressure, but you don't have enough pressure during prestart to open it, so it doesn't allow any lox to go to the hex. Line to the LPOT is all filled, it stops at the MOV. It also feeds through the lines leading to the preburners and stops at the preburner valves. At the furthest point in the lox system, right before the FPOV you have the lox bleed valve that keeps lox flowing through the system, running off to the lox pond where it boils off. This is especially important during flight, because if you form a big lox bubble and it collapses, the bottom of the tank falls out, and you get ????. MFV is downstream, the two skin temps are on the outside of the line and all that they are doing is assuring that we don't have any leakage at the MFV. If we do you will see this

thing come down (referring to the pid on plot #1). We have what are called LCC's, Launch Commit Criteria, limits that are preset, that you can't violate in prestart and still allow the engine to start. If this MFV DS temp falls below 260, we say stop, hold it, we have got a problem. It typically means that you are leaking fuel past the MFV. So what happens here is that we drop fuel into the engine, and you see the thing start to cool down. It's not cooling that dramatically, just look at the scale, and the time scale is about t-8000sec. We have to drop lox at 1.5 hrs before the test and we drop fuel 1 hr before the test. You can see the cold fuel sitting on the other side of the valve. The jumps in this plot are because in PSN3 you do a He purge on the other side of the valve. The He warms the area and that is why you get the bump. This purge comes in for 3 minutes every hour.

When we go into PSN4, which is the last before engine start, we purge this area continuously. PSN1 and 2 are just a couple of minutes each, they are very rough purges of the engine to get it ready. A lot of times we don't even take data during these two purges. We don't have any cryogenics on the engine, we're just pumping He and N2 through the engine. I will get you the documentation of when and where the purges occur. A lot of times, we don't start taking data until we reach PSN3.

During PSN3, we do several things to the engine: 1, purge the HPOP Intermediate Seal. We never stop purging this seal, because it is the most dangerous point of the engine. You have lox on one side and H2 rich gas on the other side. If the two come together, you have problems. We purge it with N2 in PSN3 and He in PSN4. The reason that we switch is that N2 is cheaper.

Jeff: Do your purges follow a rigid schedule like in flight?

Erik: No. In flight, everything from t-9 minutes on follows a very rigid schedule, because they are all controlled by the ground computer. On ground test, the only thing set is from when start enable is declared, which occurs at -t-4 sec. Everything else is floating. PSN4 usually takes anywhere from 4-12 minutes. PSN3 usually takes about an hour.

Someone: What determines those times?

Erik: First of all, you have to have lox on the engine for an hour and a half, and you have to have fuel on it for an hour. These are minimum requirements. After that, it's simply a matter of when they are ready to start. We have 4 minutes for PSN4 because that is what we do in flight. I have seen PSN4 last as long as 15 minutes. On flight, you can be in PSN4 for a maximum of 12 minutes, because of the limited amount of hydrazine to fuel the APU's. On flight, PSN3 can last 8, 10, 12 hours. If you sit in PSN3 long enough, you see these temps stabilize.

Next is the OPOV skin temps, and this measurement is actually upstream of the OPOV. You have a small gaseous bubble in this

line, and if the OPOV is leaking this temp will go down because that bubble will disappear and bring the lox into contact with the wall.

The AFV downstream temp, remember this is the little valve on the line that goes to the hex, and it stops flow from going into the engine prestart. This measurement is downstream of that valve to make sure we don't leak any lox to the hex during prestart. If you leak lox past any of these, it would be an instant disqualification of the test, because it would be an indication that the engine was not properly prepared for the test. You see that it is trending down, but look at the scale and it's not that big of a drop. 0 degrees is not cold on this stuff.

Catherine: What kinds of temps do you see if you do have a lox leak?

Erik: If you have a lox leak, you will see temps in the range of 170, 180 degrees R. If you have a fuel leak, you will see your temps fall below 200 degrees R. All of these measurements have LCC limits which are preset. You get below these, you can't go on.

Do you want me to go over the OPOV chart again? (The answer was yes). OK, we have a bubble that sits upstream of the OPOV because its stagnant flow in that area, that if you have a leak in the OPOV, you will see this bubble disappear and the temps drop dramatically.

Next, we have our fuel turbine and our lox turbine temps during prestart. Remember that all of our time scales in prestart are typically 0 to -8000 seconds. You notice that the trend is for these temps to come up. These temps are located on the discharge of the two turbines. They are there to check for a liftoff seal leak. These temps rise because of the heated N2 purge which is going on at the beginning of PSN3. It goes through all of the Hot gas system, and it is done with N2 that is heated about 90 degrees. We do that to dry out the water. Must get all moisture out, because if cryogenics come in contact with the moisture, it will instantly freeze and can cause problems. We turn it off and you see the temps fall. In the HPFTP, you have a liftoff seal that separates the pump end from the turbine end. During operation, we're cooling the turbine end with H2. That H2 is coming in through the pump end. We can't allow that H2 to enter the hot gas area during prestart so we have this liftoff seal, a spring loaded seal. If it fails, i.e. opens, you see these turbine temps drop very quickly. That is what we are looking for on this graph.

Tim: One question I have is why aren't there any comparison tests plotted on these prestart plots?

Erik: Because all we're comparing to in prestart are the LCC's. A lot of the things that occur during prestart are determined by the guys in the stand, not by the engine. It's hard to compare to another test because the guys may have been doing different things

during that last test.

Next is the lox dome temp, it is checking for leaks past the MOV. Remember I said that during prestart, the lox flows up to the MOV, and there should not be any lox past that valve. This temp sensor is located past that valve, and if you get leakage past that valve, you will see those temps drop very quickly. This rise is the result of the heated N2 purge, and you see it drop very quickly, because once you turn the heat off, the temp of the N2 drops very quickly. And at the tail of this graph, you see where we go into PSN4. During this purge, we turn on a fuel system He purge. Once again, we're looking for a quick drop off. That would indicate to us that we have a leak past that valve.

Now, we have ambient powerhead temps. These are to take temp measurements around the engine, and they are there looking for fires. We have 12 of these. They are of two types, the first are glowplug measurements. 4 of these are situated around the top of the powerhead. They are temp measures with spark ignitors located next to them (just like spark plugs in your car). It serves the same purpose as a spark ignitor on a gas grill, it burns all the gas in the area. We do this so if there is any loose H2 or lox, it will burn it right away without letting it accumulate and cause a major fire. It consequently allows us to check for leaks. The glowplug temps will go up in case of a fire. The ambient powerhead temps will drop, because you are pouring lox and LH2 on them. Prestart we're primarily looking for leaks, mainstage we're primarily looking for fires. We have LCC's during prestart and redlines during mainstage to limit these measurements.

Claudia: Could you clarify on what happens in case of a leak?

Erik: You see glowplug temps go up because of the flame, you see the 8 ambient powerhead temps drop because of the presence of lox or liquid H2 on those sensors. Either situation is reason for disqualification.

Tim: We have 12 measurements, right?

Erik: Yes, 4 are the facility glowplugs that sit about 15 ft away. 8 are fitted into the engine framework, sitting within inches of the engine, the ambient powerhead temps. You can get a substantial fire around an engine. On A2 they burned the stand, a large fire caused by a nozzle leak. There was an aluminum grate, used as a platform that was pulled back with a 7 foot diameter hole burned in it. We're talking hot fires.

This sensor pictured here is a little erratic. This may be an indication the sensor is going bad. Or, for this particular measurement, it could mean that the wind has shifted. In either case, this would raise a flag in your mind that you would carry over to mainstage, remembering that this sensor looked funny during prestart. These spikes and shifts that you see aren't bad, they only represent a few degrees. What we look for are major shifts in

the data. For example, if you see them come down to liquid H2 or lox level, then you would start inquiring, 160 200 degrees. For fires, you would see them go to 680, 700 degrees. Some of these shifts are just wind caused. Also during pretest, there is alot of liquid air dripping. It will drip on to a sensor location and cause a shift, especially like the LPFP's that have cracks in the insulation. You run LH2 at 40 degrees through these pumps. Air liquefies at 190 degrees. So we have temp to liquify air.

Next is the fuel press interface temp, it acts as a protection of the engine from the facility. We have had cases where the facility would leak fuel into the engine. This graph has a very small scale, it looks ok. You are looking for shifts to indicate leakage.

Next is the HPOP Intermediate Seal pressure. This is the only thing which is purged continuously during both prestart and post shutdown. It is the primary buffer between the lox pump and the lox turbine. The lox pump is carrying lox and the turbine is carrying H2 rich gas. The spike is common. It is the indication of a bad data point. We're purging with N2. At the tail is the shift that indicates we're going into PSN4. We purge with He at a higher pressure during PSN4, that's why you see the shift. This measurement has a lower and upper limit.

Next is the fuel system purge pressure. It is a He purge for 3 min of every hour. The rest of the time it is reading ambient. During the purge, it will read between 300 and 400. This is a difficult check to do. Basically, it is done by a PCA, pneumatic controller assembly, that checks for leaks. This plot tells you if you are getting through the PCA correctly.

Next the lox preburner pressure. The s/d stands for shutdown because that is where we normally use it. We turn the purge on to get the lox out of the dome, because if you don't, you get pops which are energy burst. They can cause damage. In two of the last three A1 tests, there were pops in the dome that were physically bending the faceplates.

Tim: We are still in prestart right?

Erik: Yes, we monitor the purge here. It is limited by an LCC. If there is a problem here, it will automatically generate a fid. When you go from purge 3 to 4 you switch from N2 to He purge. On this measurement, you get the shift before cutoff because at 3 minutes prior to cutoff you started the 3 minute He purge on the EPOP Intermediate Seal which caused backpressure in this measurement, thus the shift up.

Let's go over it again. When we begin PSN4, we manually turn this purge off. You are normally sending N2 to the OPB and FPB purge pressures and the HPOP IS. But when you switch to He on the HPOP IS, you have more N2 to go to the two preburners, so you get backpressure and thus the shift that shows up here.

Next, you see the same response on the FPB purge pressure. Your levels are different, I don't know why. I think we are being eyeball fooled by the scales. It levels out on this graph because you have reached thermal stabilization. Remember that you dropped propellants, and you also had LN2 going through.

OK, here is your emergency shutdown pressure. We have two ways to shutdown the engine: hydraulically (normally manipulate all the valves this way.) We have a hydraulic component, hydraulic return. We have hydraulics running throughout, we shuttle it to different cavities and to actuators, and that modulates how you turn the valves around. If we get into a problem with hydraulics, i.e., APU fails during flight, we go to the phase in the controller referred to as hydraulic lockup. We have tested that on A1 in several tests recently. In this phase, the engine recognizes that it can't control anymore, it just leaves the valves where they are. This is an intermediate step between normal operation and shutdown of the engine. We translate the engine from normal operation to fixed orifice mode. This means you don't have any controlling movement on the 2 preburner valves anymore, so your not running the engine very efficiently. This leads to a pneumatic shutdown which means that we use the He on the orbiter to drive the valves closed. We monitor that with this Emergency Shutdown pressure. This measures the pneumatic pressure available to shut down valves if necessary. In prestart we are making sure that the valves are closed. The hydraulics and the He are closing the valves. The driver is a servo-piston arrangement in the valve, one side driven by hydraulics, the other driven by He.

We can immediately identify PSN4 on this chart, because we take the He off of that valve. The other blip is the fuel purge, remember the one that comes up 3 min of every hour. Because the He is taken away to drive the other purge, you get the drop on this measurement. Refer to the 1 pg purge schematic. The large blip would be noted as an anomaly, but in reality, we know from experience that they are manipulating He from the stand. Typical scenario, "what is causing that?" One, downstream in the PCA, or upstream in the He interface pressure, He coming from stand. Look at the upstream measurement, it's doing the same thing, surmise the facility is causing the blip. Experience tells us that is nothing. This would be an anomaly of little significance, but an anomaly none the less.

This next plot is 0 to -10 sec, showing that HPFP inlet pr is within the box it needs to be to start the engine. These arrows indicate the LCC.

Same thing for HPFP in Temp. (He was pointing at something in this plot that I don't know about, so this entire discussion was useless) Why do we use these if we need those? Because we don't have those on flight. We don't have engine measurements on flight. The measurements we have are orbiter measurements and it's instrumentation is poor. So we use these measurements instead.

In case you haven't noticed, any pid number below 300 is a cads pid, anything above is facility. We don't have facility pids during flight.

Tim: On these inlet temps, would you ever see any changes that tell you anything?

Erik: Yes. If you get changes here, you get outside of the LCC and the test would be stopped.

June: Do the rates change on these graphs?

Erik: Yes, they do this often, I don't know why. Sometimes they even switch the data system off for a minute. I don't know why. We get straight lines from that. They have done it when they turned the nitrogen off.

Amy: Do they do this on every test?

Erik: Yes, I think that they are exercising the different data rates on the data recorder to insure it's integrity.

On the fuel side we do one pressurization, drop fuel giving a small amt of head (the weight of the fuel itself), a couple of psi, then they pressurize the ullage (the gaseous part of the fuel in the tank) to bring the fuel pressure up to where it needs to be.

On the lox side, they do two pressurizations. They drop lox, which gives a large amount of head, and then they pressurize once early in the test, and then they pressurize here (where you see the big jump on the HPOP inlet), just a little before the test. Do this a couple of minutes before the test. You are sitting here in PSN4. When you pressurize and bring all of the parameters into range, you declare engine ready (the controller does). You can't start from purge 4. You have to be in engine ready to indicate that you have the lox pressurized, the cryogenics and He in the condition they need to be to start the engine. One more phase, start enable at t-4 sec., to open pogo valve and close two bleed valves. You can't leave them open because the HPOP discharge pressure is so high that it would blow off the downstream duct. We have done tests in the past where we left the bleed valves open and put shims in to absorb the pressure, but the shims were crushed.

Next is the Lox inlet temp. This measurement only has a lower LCC. The PBP discharge temp downstream is the upper LCC. Reason: lox coming through becomes restricted, it will heat up because of the ambient conditions. Many times on ground test, we will start below this LCC, especially on B1. We know we can do it, so we do.

PBP DS temp, we don't typically get near that upper LCC.

He inlet pressure. The drop is caused by initiation of PSN4. Remember, for PSN4, we switch from ground N2 to He. This drop occurs because He is now being required to feed the Intermediate

seal as well.

Next plot shows where they turn off GN2. The little jump is where they turn off the purge to the intermediate seal, the upswing is driving the two preburner pressures. Then they cut it off. By the way keep absolute and gauge pressures straight, because they will often switch between measurements.

Hydraulic supply pressure, note two things: level and scale. The two blips you see are them fooling around with the facility. Certain things going on in facility we don't understand because we don't need to. You see the high levels that they use to drive the valves.

MFV Hydraulic temp. Make sure these temps stay warm. You freeze hydraulics at about 380R. Spec says you have to operate at 420 or above. We know these numbers, because we rerouted lines on A1 that have caused freezes and consequently, we have had problems shutting down the engines. We have gotten severe pops, faceplate damage. Also remember that hydraulics are going through the actuator that is attached to valve that has 40R fuel sitting up against it. We have a heater in the actuator to keep the hydraulic from freezing up.

This next plot is the most confusing. We have 4 parameters from one test. It is a very short time range. Engine status word tells you exactly where you are in your test. The first spike is start enable, about t-3.1 sec. Then you change the position on three valves. You close the lox and fuel bleed valves, and open the pogo riv (recirculation isolation valve). It prepares the pogo so that when you pump gaseous oxygen through it, it will release. It goes from 0 to 95. It was supposed to go to 100. Perhaps this is something you would flag. It is obvious that it is full open because it flattened out. It may be miscalibrated. You are looking to insure it went from full closed to full open in the allotted time frame. This time frame is 2 sec. This applies to the bleed valves, they must go from full open to less than 20% in that 2 sec time frame. The lox bleed valve closes no problem, but the fuel bleed valve, we have a problem with the rvdt that reacts too slow. We had a no start on STS-26 because of this. The bleed valve itself was ok, but the rvdt was too slow to react. rvdt - rotary variable differential transducer? rvdt's and lvdt's measure position, rvdt does it rotary, lvdt does it linear.

Pogo precharge pressure - we precharge it here. When you start an engine you can get two kinds of pulses through the engine, pogo and screech. Wave starts in the engine that matches the natural frequency of the HPOP, and it starts a wavefront that starts banging the engine. Low pulsing - pogo, high pulsing - screeching. Pogo is dangerous because it can match the natural frequency of the struts that attach the ET to the orbiter. Pogo has half lox and half ullage of gas, prestart is He, after start is gox. The gox comes from the hex, it splits between the repress line and pogo ullage. It decouples the whole system, pulse comes into the pogo,

acts as a shock absorber? Right before start, we charge it with He. You have to have ullage for the pogo to function properly.

Tim: Why does the pressure drop if the pogo is charged?

Erik: Because it is measuring the pressure in the line, not the pressure in the pogo.

June: Is the time important?

Erik: Yes, but its dictated by the controller.

The next few plots for prestart have comparisons to other tests. You see the fuel inlet pressure. Note the small scale, and the measurements are very close.

Tim: Do you use the same comparison tests for all comparison plots?

Erik: No, you choose different ones for different phases, usually. For prestart, you choose comparison tests on the same facility, because the facility drives how you precondition the engine. During mainstage, you look to match up profiles, engines, pumps, etc.

Lox inlet pressure - your comparing inlet conditions across tests. Between t-4 and start you will see some wild variations, but note the scale. In the facility, you are shuttling valves back and forth to prepare the facility to deliver fuel and lox properly to the engine. You don't see it in flight, but it's not detrimental. You see how repeatable it is between tests. It's all computer controlled.

Lox inlet temp - wild variations, but notice the scale.

HPOP intermediate seal pressure - once again making sure that you have more than enough flow going to the intermediate seal. The blip here is the result of pogo precharge pressure. If you look at the fuel purge pressure at the same time, you see the same kind of drop.

Next, He interface pressure - this variation is the results of the pogo coming up. You have a regulator to keep this constant, but because so much He goes to the pogo, it can't keep up and you get this drop with a quick recovery.

End of prestart.

The other time that we may look at prestart is if we have leaks in prestart, it may affect things in mainstage. Also, if we appear to have bad instrumentation during mainstage, we'll go back and look in prestart to see if it was biased then.

Tim: Are you talking about plotting additional data?

A3-69

Erik: Same or additional. For example, HPFP discharge pressure may appear to be reading 50-100 psi lower than expected. First question is "Is it real?" The easiest way to check that is for bias in the prestart data. If it's reading low during prestart, then you have an instrumentation problem, not an engine problem.

Claudia: Are you reasoning about those two separately or at the same time? It sounds like you are reasoning about instrumentation and engine at the same time.

Erik: Same time. We first try to eliminate the instrumentation problem. You don't want to waste time chasing problem that were really instrumentation.

June: During prestart, do you have measurements for every sensor?

Erik: We get all of the data. The only thing we may not get is scan rate. Facility at 50 cuts/sec. Cads at 25 cuts/sec. We get readings from every piece of instrumentation, but we may not get it at the same rate that we receive it during mainstage.

June: So you are getting garbage data and you know it?

Erik: It's not garbage data. There are situations where you use it. There is some problem that you need a measurement and it turns out that it is there.

Let's go over start. Two ways to look at start: comparison to other tests, compare to 2 sigma database.

Lets start with the comparison to other tests.

Tim: What are the differences between the two types of information that you get from these different comparisons?

Erik: 2 sigma tells you if you've been there before. If the start is between the 2 sigma, we assign it a much lower priority in terms of danger to the engine. We've started the engine with those parameters before and it started safely. Failed starts are compiled in another database. Second, we look to see if we're starting the lox side hard or easy. Hard starts (hot, slow, quick, etc) are defined by lox or fuel side coming up fast. Whether you get the engine to mainstage in 4 or 4.5 sec doesn't matter on the ground. What matters is that all parts of the engine interact properly, i.e., don't start the lox side so fast that it stalls the fuel side or vice versa. Our main indicator for these is prime times. Priming is when you get full flow across a faceplate. You prime the fuel preburner, main chamber then lox preburner. You have to maintain that order and the deltas between the times that they occur. For example, if you prime the main chamber too close behind the fuel preburner, you may stall the fuel pump. When you do that, you stop fuel flow, go lox rich and lose the engine.

You have to have a regimented way to bring up the different parts of the engine, to bring it up safely. You don't want pops or stalls.

Look at MCC Pc - multiple tests plotted. Look at differences between the tests. These are different tests from what were used during the prestart comparisons. Example of using different tests is if you do a software change during start. Valves are run open loop, closed/opened loop, and completely closed loop. We will make software changes on how we modulate the valves during start to try to get a smoother, safer transient. For these comparisons, you want tests that had the same software. You may have different comparisons for mainstage to match profile, engine, stand or pumps. All comparisons do is to give us an eyeball.

Pc comes up nicely. Two main control phases during start: 0-.74 sec open loop control, at .74 sec we begin Pc control to ensure that we're controlling to Pc Ref. At 2.4 sec we go to mixture ratio control, mainstage, when we start controlling fuel flow. Up to about 3 or 3.5 seconds, the valves are very repeatable. At 4 sec and on, the valves begin to deviate to control the engine. One of the controlling factors is Pc, that is why Pc is so tight out here in mainstage, because it's controlling OPOV position.

Next is ICD limits for this test. Not many ICD limits. We do have MCF's, that you can think of as redline limits during start. We have them to help on flight. We start engines 6.6 sec before launch. Up until then you have capability for on pad abort. There are limits on measurements, if they're exceeded you do abort. You see these limits on this plot.

Tim: On the Pc chart, you see variations. How do you reason on that?

Erik: This is where the comparisons come into play. They tell you the history of the engine on Pc. Has it done this before? How did we analyze it? We don't redo those analyses. If it was safe, it's ok. We may flag an aberration as an anomaly, but we know its a safe one. If you see variation past start, then you may have serious problems.

Oh, by the way, we usually start to RPL(100%). We have started to other power levels, but it's rare. For the P&W pump, we started to 65% because they didn't feel safe starting to 100% with the new pump.

June: Are you saying that the variations on this test are normal?

Erik: Let me show you something. We determine fuel preburner primed when we see a predetermined change in slope on fuel pump speed. You are now getting real power to turbine. The main chamber priming is when you see this big rise on this Pc plot. It is defined as primed when the MCC Pc breaks 100 psi.

A3-71

FPOV command - very repeatable, open loop. Conditions are determined in software spec. Valves are trying to get to desired Pc Ref. FPOV is trying to maintain a certain amount of fuel flow. For different pumps, you need more or less fuel to get the same amount of flow out of your turbine.

Same thing with OPOV - very repeatable up to about 4 sec. This one is incredibly repeatable considering that we changed all of the pumps. We changed out 3 of 4 pumps between this and the last test.

Next is OPOV and OPOV command. OPOV command keeps the engine from running away on the lox side. If we have some bias that tries to open the OPOV by 4% or greater, this limit prevents that. At a given time it looks at the interaction of the OPOV and MCC PC and determines OPOV command limit from those. This is very important during mainstage. If you have some sort of instrumentation problem, you don't want your OPOV running away. If the engine still has problems, it will more than likely shut down from a redline on the lox turbine temps. See the little bump here, we incorporate that for transient overshoot. We know that we get it, so we account for it here.

Tim: Is it important for us to know all of these little variations in FPOV and OPOV during start?

Erik: I wouldn't get too concerned, the 2 sigma package shows that is ok. If you are significantly off during the later portion, that being 4 sec on, you want to look at it. If you're off by any thing in the early portion, you have to look at it, you have a problem. Or, you have done something with the software. Go back to OPOV command. If you have variation of 1%, you have either done something to the software or you have a problem, software is wrong, data is not being read correctly, or you have an anomaly. Different phases of start cause different concerns associated with them.

Go on to HPFP discharge pressure. From .8 to 1.4 sec we are looking for fuel side oscillation. Fuel pump discharge is before MFV. We think that the 40 degree fuel is coming down and hitting the hot nozzle, it gasifies and forms a pressure bubble that comes back here(?) and stops this flow for a second, and that is the pressure that you see here. It affects the way we see the engine start. When you get a fuel side oscillation, you usually see a harder start, hotter lox turbine temps, because you instantaneously rob fuel from the engine which increases your mixture ratio and that results in a hotter start. This plot shows two tests with a good size fuel side oscillation, the triangles and the squares. By oscillation I mean that it goes up, drops off and then starts to go up again. The current test plotted doesn't show it. The pressure levels but it doesn't drop before it begins to rise again. We are missing fuel pump discharge temp, but if you look at it at the time of the oscillation, you will see a 10-12 degree spike. It is a better indicator of the fuel side oscillation. We can't control

it, we don't know why we get it! It's not necessarily a problem,
it's just that we can't predict or control it.

Erik Sander (continued)

Start Performance Package, (referenced by plot # on upper LHS)

Q: if you had a hotter start, if you noticed you had a fuel side oscillation, what would you expect to see during mainstage?

A: No difference, it all washes out by the time you get to mainstage.

Q: Does it take life away? Why do you analyze this?

A: It's something we analyze in terms of how hard the engine starts. We have software parameters that we can use to control how hard the two different systems start - the fuel system and the lox system. We have to use those because we have to bring up the two different systems consistently and within certain prime time boxes - make sure the engine starts properly. If you're on the hairy edge of one of those boxes, a fuel side oscillation may drive you over the hairy edge. These are adjustments we make post-test. Want to make sure we have enough room on our limits so a fuel-side oscillation doesn't drive us into a problem area.

Q: So if you observe a fuel-side oscillation, do you want to slow down the fuel side so that it matches the lox side?

A Example: Suppose we had a hot start w/o a fuel-side oscillation. We're on the edge of determining whether or not we want to recommend a software change to make the start a little cooler - there are knobs that can be tweaked to make the start a little cooler. The fact that you did not have a fuel-side oscillation may drive you to making that recommendation, because you know a fuel-side oscillation will drive you even hotter. This is an example where a fuel-side oscillation may drive the recommendations made, even though we have no control over it.

PLOT #7

Here are the fuel turbine temps - A and B channels. You can initially see some oscillations and then you continue on through the start. Notice that we're very similar to a previous test (two tests back) even though we had a different pump. On the lox side you'll see a very similar trace - even though we had a different fuel pump and lox pump and everything else. We look at the repeatability with the comparison tests and look at the repeatability with the 2-sigma database to make sure you weren't way out of line with all three tests. As long as you're within that (we do have variations within the start) we determine we have a good start.

Q: Even though you're off by an amount towards mainstage, you'd still call that an ok start?

A: Yes, and the reason is that you have different pumps here. Two different big pumps with different efficiencies. What we typically do is note this during start and keep it in mind to look during mainstage and see how it was reacting there.

During start, we're particularly concerned with this parameter, because there's an MCF (redline between engine start and launch) which states that at a certain time (around 4.98 sec) the fuel turbine temps have to be under a certain limit. You can actually shutdown on the pad because of that. We've shut down ground tests because of it.

Primarily what we're doing here is noting differences and carrying them to mainstage. You have to have a memory between the phases - system will need one too. You need to remember differences during mainstage and as you transition to cutoff. If turbine temps are 70 deg higher here, the first thing you'll look for in cutoff is if the temps are 70 deg higher there too.

A2-75

Q: Does the severity of the fuel side oscillation impact at all where the turbine temps level out?

A: No. Get back to this in two slides

Plot #8

Here is the B channel -same type of effects.

Plot #9

Good example of effect of fuel side oscillation at around 2 sec mark. Remember on the other two tests we saw that a fuel side oscillation gives you slightly hotter lox turbine temps, especially in the beginning, because you're robbing the fuel from the preburner and raising the MR. By the time you get up in the later ranges, what's determining these turbine temps is primarily the efficiency of the pumps and the balance in the system, those types of things.

Basically, anywhere from four seconds on, your prime determinant of what's happening in the system is what the controller is telling it. What the controller is telling translates thru pump efficiencies and line resistances, etc. into the data that you see.

Plot #10

Here is the other channel and you can see the early effect of the fuel-side oscillation (or lack thereof on this test). Slightly cooler temps because you're not robbing the fuel from the preburners.

Plot #11

HPFP discharge pressure - again, looking for general repeatability. This parameter is different from the turbine discharge temps in that it is more repeatable towards mainstage than early in profile. This parameter is driven primarily by engine resistance. You have a certain amount of fuel flow that you're trying through drive to a downstream resistance. You've got the same engine. (You do have different turbines in the preburners that partially determine resistance, but one of the primary determinants is the engine and you've got the same engine. That's why it's so repeatable.)

Q: As you continue to test an engine and it becomes leaky, would that show up here?

A: Yes. The analyst has in his mind a lot of things that drive this parameter. You know there are some gains that say if you change these pumps out, you change the turbines and therefore the effective resistance since each turbine has its own resistance. Work your way down the line. If you do things like leak the nozzle or crack the chamber, (you have fuel pump discharge branches feeding directly into the nozzle and directly into the chamber) you lower the downstream resistance. Very important to keep in mind. The engine is controlling to that flowrate (where engine flowmeter is) on the fuel side. What happens downstream, the controller doesn't know. You could be dumping all your fuel out the nozzle and it wouldn't know. It will just sit there and happily run along to that given fuel flow. It doesn't know what gets into the chamber; it's not scheduled to know it. When you have a nozzle leak, you see the MR go through the roof. We did that on an A1 test. It was sitting there fat, dumb and happy running a certain amount of fuel. Meanwhile a considerable amount was leaking through the nozzle. You lose fuel that was going into the chamber. Other side realizes it doesn't have enough chamber pressure and powers up the lox side. That's why the MR was going through the roof. Controller does nothing to keep up HPFP DS P, it will let that pressure keep dropping as you leak fuel downstream.

Plot #12

HPFP Speed. No speed on this test! The other two tests do show speed but there's a facility filtering problem on 902-554 - doesn't pick up the speed until it gets to a certain level. They've seen this before. Is it an anomaly or is it instrumentation? In this case you have other things telling you right off the bat that this can't be real. You have HPFP DS P coming up. How can you get pressure without speed? Obviously instrumentation problem. SSC verified that it was instrumentation problem. We do have a CADS backup to this parameter which they did in fact look at for this test. They found post-test that the torque on this pump was very high. They did go in to look at the CADS speed and compared it to the CADS speeds of other tests. Since they could barely turn the pump after this test, you want to look at the data to see if there's an indication in the data as to when that happened - start, mainstage, shutdown. If it had happened during start and it was strong enough, you'd expect it to slow the pump down and you'd need a lot more valve position to get the pump speed where it should be. Or maybe you'd see it come down a lot faster during shutdown. This is a case where a HW inspection revealed an anomaly and you were glad you had backup for failed instrumentation. If not, you'd look at other things such as the pump discharge pressure.

Plot #13

HPOP discharge pressure. Changes early in profile are due to efficiency variations. This parameter is driven by two things only: the resistance of the main injector and how much lox you have going through. Very repeatable near mainstage since you have the same engine and are trying to pump the same amount of lox.

Plot #14

LPFP speed. See some variations, but we have different pumps.

Plot #15

HPFP inlet pressure. Driven by two primary things: engine inlet pressure, is it repeatable? and the LPFP speed. Note on the previous plot LPFP speed was a little low for test 554. If it were very low, you'd wonder if it was instrumentation or real. So you look to HPFP inlet pressure; since it is also low LPFP speed low is real. If it were a greenrun test, you'd check to see if it were acceptable.

Plot #16

LPOP speed. Driven by HPOP DS pressure. This pump is different than the rest of them since the turbine inlet and the pump inlet both exit through the pump discharge. You can do that on the lox side because everything is liquid.

Plot #17

HPOP inlet pressure - just LPOP discharge. Very repeatable - no problems there.

Plot #18

POGO precharge pressure. Because of the way you reconfigure when you go into M/S, you get a reconfiguration of the valves and this pressure is actually reading the POGO ullage now. Here you see the pressure come up, it gets another shot of helium (around 2.4 sec), maintain it and then the LPOP discharge pressure takes over. Want to maintain the ullage during start.

Plot #19

HX interface pressure. HX turns lox to gox to go to the repress. There are actually several measurements in that line. There's a CADS measurement that's called HX discharge or something like that. Lox comes out of the pump, goes thru the HX, gassifies then goes to the orbiter or pond.

Plot #20

HX interface temp - just a tap in the same system. Notice we start about ambient. Now we have the AFV prestart to keep any lox from coming into the HX - that's why it's reading about ambient at start. Pressure starts to build and at about 140 psi the valve pops open. Drive lox into system and don't have significant heat in the burner and that's why this T takes a nosedive. You start getting a flame and that brings it back, and then you start getting significant heat - truly gassified because you have the hot temps. Good bit of variation at the start of plot because you have different temp days, lox down there different amounts of time, etc.

Min and max flowrates on represses. Min is 1.1 lb/sec and max is 2.35 lb/sec at RPL. There's a valve system downstream of this temp that you're opening and closing to vary repress flows. That won't significantly effect temp you see here but it will significantly effect pressure. If you see a pressure going high or low the first thing you ask is if you're comparing to a test with the same repress conditions.

Plot #21

AFV position. This tells you it opened. This is another LCC. All these have to be met to launch. If you don't have lox flowing through there, you'll burn up in nothing flat.

Plot #22

HPOP intermediate seal purge pressure. Slight dropoff around 2.4 sec is when you charge the POGO. Another spike later (around 4.5) when POGO precharge turns off.

Plot #23

Preburner purge pressures. Purges in two preburner domes. Make sure they're off during start. Turned nitrogen off in purge sequence 4. Spike later on is a reaction to POGO going off - see it every test, it's normal.

Plot #24

Fuel purge pressure. Want to make sure it's off - that you're not trying to put any purge in there during the start.

--END OF START--

--BEGINNING OF MAINSTAGE--

There are certain facets of the analysis that you lock into your memory for mainstage. Prestart- what kind of biases did we see in the instrumentation, start- how does it look like we transitioned into mainstage. If you see a mainstage problem, you want to keep these things in mind.

Several mainstage packages : GRMAIN, an instrumentation package where we look mainly at redundancies, and DATAACC which is the basis of the presentation package we do for Otto and the rest of them.

Plot #1

Current test pc and controller reference - 5 sec and on. Start to RPL, go to 104, and then bucket down to 65, 64 and 63. On missions right now we only bucket down to 70%. We do this to clear the preburner pump of bistability. Come back up to 104%, throttle up to 109% and then a slow throttle down to 65 again - this is to simulate the 3g throttle in flight, go back up to 100 and the S/D from 100. They didn't use to have to S/D from 100, but this reduces the chance of having bad pops in the preburner. In flight we S/D from 65.

This is called a greenrun profile. Also called a calibration

test. If your primary objective is certifying the engine, it's a calibration test. The primary objective is to run it in a consistent mode with how you've run it before so you can compare and also to calibrate software constants. Final test in acceptance series is acceptance test and this is primarily a mission simulation. Greenrun test - greenrunning pumps. Has to be run through this series of pl transitions and venting conditions to certify it for flight. Run pumps thru flight conditions.

This parameter you're making sure you're following a pre-coded reference. Controller translates reference pressure into valve positions. Looking for spikes (instrumentation problem with a pc transducer), wandering (hydraulic lockup), and spikes in bucket where you're checking for preburner pump bistability.

--BREAK FOR START 2 SIGMA--

Shows mean and plus and minus 2 sigma values. There is no test data on any of the plots in this package. Valve positions are tight early on because it's open-loop control. Spreads later on in reaction to different pumps and engines that are being run.

Q: How do you select what you include in the average and two-sigma calculations?

A: We go thru a bunch of different starts and see whether it's nominal in terms of software (look at pre-test). Also, is it relatively nominal in terms of M/S MR. Everything else they pretty much keep in there. They're up to about 160 firings.

Q: If the data falls within the bands is it automatically ok and if it falls outside it is not ok?

A: There are no regions that are just plain ok or not ok. If you're well within the band you've got experience behind you saying you've operated there before w/o a problem. If you're right on band, then you know you're in the 10% range of engines and you begin looking for clues as to why you're out there. Look at other related parameters. Let's say you had an FPOV position that was really high. Know historically that this CADS measurement is usually good. What's causing it? Could be caused by bad fuel turbine.

Why is the FPOV so high - there are several different drivers. You're interested in pumping the correct amount of fuel flow. Is the flow unusually high - check other plots - no, flow is ok and MR is right on, therefore c2 is nominal. To drive flow you have to supply power to the turbine which turns the pump which drives the flow. The next question might be do I have a real bad pump or turbine? Look at turbine temps - they're high. This confirms that the valve is high - could be that turbine is bad. Now check discharge pressure - is it low? This is being supplied by the preburner pump, so now there is a lox system component that's feeding over and you're seeing the effect in the fuel system. If turbine temps were very low and FPOV very high that would indicate that turbine and pump are ok but PBP discharge pressure is so low that you have to open up valve to get correct amount of flow. Have seen this several times - PBP discharge pressure is so low that valves have to scream open to get you there. How much of a problem that is depends on a lot of things: if you get 3 or 4 sigma high with this valve (FPOV?), you have a good chance of running into its limit. This has happened before. Open it way up and you still don't get enough flow. Valve - turn it and open up window to the flow. As you get to the end of the range, there's not much window left. By doing the last 10% there's not much resistance change, you don't get much out of it, the valve will scream open and you'll be sitting there without enough fuel flow and controller can't control properly. You'll run into high MR situation.

If you conclude that turbine is pretty bad, you go back in history and look at previous run of this turbine on another engine and confirm that it ran hot there too.

That's how you use these things. It's not as cut and dried as if it's outside it's a problem, if it's inside it's not. If we see it go from minus 2 sigma on one test to average or plus one or two sigma on the next test, it's still within the band but it definitely indicates a change. Then you check to see if you changed the pump or the software, etc.

Q: How would you know if it went from minus to plus two sigma? Are both tests on the same plot?

A: No, they're not. You have to rely on the memory of the data analyst. That's why there's one analyst per stand. That gives him a history of the stand and typically a history of that engine in his mind.

AN ASIDE
--MAINSTAGE TWO SIGMA--
(not in our plot packages)

During mainstage we also have a two sigma comparison. They have a program which compares temps, pressures at certain conditions to a database average and updates the average, etc. This is in a table format. It gives you temps, pressures, speeds, etc. along with the averages, one sigmas (which get updated automatically), and the sigma that you're at in this test. This is a good mainstage eyeball of what you just saw in the start two sigma plots. In the example shown, the HPOP DS pressure was 4.5 sigma high. This is a definite indication to go back and find out what was going on. On this engine there is a history of a very high resistance injector. A weld was offset, they didn't iron it out and therefore a high HPOP ds pressure resulted. This drives other things in the system. Already get a feeling for mainstage two sigma performance at the end of start; this is a table for conditions at some point in the test. That point is max fuel turbine discharge temps in the tests. Max temps are found at 65%, 100%, 104% and 109%. (Jean wrote this program - looks for max temps and nominal lox conditions and tabulates all the data.)

HPOP DS Pressure usually means high turbine ds temps. That wasn't the case here - means pump is pretty good because hpop ds pressure is what pump has to pump against. Must have a lot of power at the pump end which is supplied by the turbine which means hot turbine temps. This table is in all the data review packages.

Going through and analyzing the data in the way that we're doing here is only the very first step and only takes the first one to two hours of time. Typically they have 1.5 days to go through their analysis. Other activities include running the programs, checking the requirements, etc. That's all part of the job and how we analyze a test. Looking at the data is not a complete job by any means.

--BACK TO START TWO SIGMA--

Plot #51

Fuel turbine temps. Start off together and end up with big spread. This is because we're running different engines, different resistances, different efficiencies on pumps, etc.

skip the rest of this package

--END OF START TWO SIGMA--

--BACK TO GRMAIN--

Plot #1

PC profile. Make sure you stay with the reference. Look for overshoots on the transients which they see a lot of - could be a pump problem, can't control the pumps properly. For the engine it's a lot easier to maintain things steady state than it is during

transients. Look for bistability.

Bistability - the preburner pump can be stable at two points of operation. Normally with a pump you have a certain flow v. head coefficient. The thing spins at a certain speed, you put a certain flow through there with resistances it will give you a certain amount of pressure rise. Some pumps can be bistable at two different pressures at a given flow; will sometimes bounce between them. Preburner pump is also called boost pump - it's attached to the same shaft as the HPOP.

CALC 127

Engine oxidizer inlet NPSP - think of it as pressure (pressure over the vapor pressure). Vented it down, pressurized it and vented back down again. Lower ullage in tank - pressurize ullage again. Must be careful of word repress - this pressurization here is also called repressurization. This is different from repress flows coming out of HX. Pump will react to venting and pressurization drastically. On ground tests we control ullage through gases (helium-fuel nitrogen-gox) that come in at the top of the tanks. On flight you see this PID take a nosedive at about 120 sec because you lose the solid rockets. Up to that point you're just accelerating. If you looked at a g-force plot, the g's go way down - the astronauts and the lox in the tank feel similar effects. So you want to run the pump at the low inlet pressure you'd see during flight. During the rest of the mission you'd see the pressure go up and then when you hit the 3-g throttle the pressure holds steady.

You see these inlet changes in the speed of the high pressure pump, and in the preburner pump discharge pressure. Recall the HPOP discharge pressure is held constant, but as you change the engine inlet conditions the high pressure pump powers up and down to hold the HPOP ds p constant. So you have a constant HPOP DS P, a varying HPOP speed, this results in a varying PBP ds p - the valves have to compensate for this. That is what you see the valves react.

(Three Info packages: an explanation of all LCC's, purge sequences and purges, and all things that happen in flight and things you see in the data.)

Back to bistability. The preburner pump usually has no problems going through all of this mess. We've seen in the past when we throttle down to 65, 64, 63, that the pump can become bistable and will actually run at two different operating conditions. The problem is that these pressure oscillations will feed all the way through down to the preburner and into the main chamber. We have a requirement that we can't have bistability in flight; so we must check on the ground if that particular pump has bistability. Some pumps do and some don't. It has to do with the inducer design. If we do hit bistability, the requirement is that you cannot operate within two percent power level of hitting bistability. For example, if we're bistable at 63, we can't operate below 65. We get around this by not flying anymore below 70%.

We power up to RPL. At 5 sec, we're still sitting on the pad. At about 6.6 sec from the first engine starting we blow off the hold-down bolts, the thing takes off, SRB's light, etc. When you clear the tower you power up to 104% (requirement). Not all missions power up to 104% - it depends on the payload weight and the orbit. When you're about 32 sec into the flight, you do the bucket - also called the max-q bucket. You're going very fast in the low atmosphere where you have high density air. There's a requirement on the tail of the orbiter that you power down; actually power down the SRB's as well. Run like that until about 67 seconds. Simply to protect the structural integrity of the tail of the orbiter. Then you power back up to 104% and just go. At 120 sec you drop off the SRB's. This does not effect power level, just engine inlet pressures. At that point you're basically at the edge of the atmosphere. You don't go up to 109% unless you get into an abort scenario. Requirement is that you not put more than

3 g's on the astronauts. Toward the end of the mission, you have a lighter vehicle (considerable portion of propellants burned), don't have to worry about atmosphere anymore, so you just GO. Right toward end you hit 3 g's so you start to throttle down. This throttle down will maintain 3 g's.

On ground test you s/d from 100% to avoid pops. Can't do that in flight.

So these are the basics of bistability - can feed into main chamber. Also look for it in other plots that aren't in the normal packages - in special packages that we build. It's just an expansion of this region and we'll plot things like PBP head ratio vs the pc.

TAPE 3 - SIDE 2

CALC 127

LOX inlet NPSP. Pressure is a function of two things - the weight of the fluid in the tank and the ullage pressure. We can't control weight of fuel, but we can control ullage pressure. We vary this inlet pressure by pressurizing and venting the ullage pressure. Traditionally have some problems when they repressurize again - could be going too fast.

CALC 126

On the fuel side, all they do is start at a high inlet pressure and they vent, don't have to repressurize. Always repressurize on lox before s/d. Have more margin on fuel pump than lox pump - lox pump cavitates more easily. Lower limit is shown on graph.

CALC 253

Here's the facility fuel mass flow in lb/sec. What you're looking for here is a nice steady fuel flow - we're holding this guy constant, pressurizing and venting shouldn't effect this. No problems in the bucket. A little hashier when it comes out of bucket but nothing out of the ordinary - very small scale. Looking for spikes, motion.

Q: What about the overshoot when it goes to 109%? And why isn't there an overshoot at other pl transitions.

A: That's how well the system can control. Always seem to get it when we go to 109%. It may be that the valve is so open that you're getting less resistance change for a given valve position change - much harder to control.

The valve is not nearly full open when you're at 109%. In fact, if you're at 90% open when you're at 109% RPL, you're in trouble because between 90% and 100% it takes very little to send it over the edge. At 90% open, you're basically out of resistance. Cannot operate at 100% open - means you're not getting enough fuel flow. You're a fixed orifice at that point since you can't open it anymore. Typically run in mid or low 80's.

CALC 219

MR. Measured by the facility meters shown with ICD requirements. Very hashy, but you can see that in general we're within the requirements. You have to be careful, we let the model guys quote the actual MR since they have all of the correction factors, etc. This gives you a general idea where you're operating. It's imp. to recognize this because if you see that your engine is running uncharacteristically high or low on one side or the other, for ex ample lox side hot, at first you might surmise you have a pump problem. But if your MR is way up you expect lox side to run hot because it has to do a lot of work to get all the lox through. This is a good indicator of how the fuel and lox sides should be interacting and running. This is calculated

parameter. Ignore PID 8 - don't use it for anything! It always says 6.01 - it only tells you what it's trying to control to.

CALC 254

LOX flow. Want to see that you're nice and steady. See a lot of hashiness - and this is a much larger scale (these are some good-sized oscillations). We see this every time we are on A2 and get to 109% - something that the system will have to learn. We think it's uninsulated lines that shoot off. Forming bubbles in uninsulated lines and when you get to 109% you lower the static pressure so much that you're sucking it down in the line. It's a previously identified anomaly - analyzed to death. They won't fix it. You're looking for shifts in lox flow. Sometimes you might see a shift, particularly in longer tests at about 300 sec. A lot of times this is due to change in repress flows. Fuel doesn't see it because it's in front of the repress flows but the lox sees it because you've lost an extra pound of lox - straight from the chamber and you have to power up a little.

CALC 88

Controller fuel flow, facility fuel flow, and difference between the two. This gives an indication of how good Kf is. We believe the facility fuel flow because we've calibrated it. So we compare controller flow to it. Typically, if we're within a pound we're happy. Otherwise we make recommendations to change Kf. One thing to look for is shifts - could be an indication of a bistable flowmeter. We've seen that happen.

Calc 8

This tells you the fuel flowmeter speed. This is the engine flowmeter and has a requirement to run under 3800 rpm at max power level. You're looking for shifts and that type of thing here.

PLOT 9

This chart may help explain the opov command limit. The topline is fpov command. If you have a pump problem or a turbine problem, you'll probably see it in the fpov. This is the control that you use to modulate the amount of power that you get. If there's a seal prob, anything that affects the parasitics, a grab on a bearing, etc., you will see it here because there will probably be a different power requirement. You see the power up, a little overshoot on the control (but the scale is very small). The level is at 80-85%. If you get up at or near 90%, you have a problem. You see a power up, the bucket, power up again, and then very little reaction to the vents. This is mostly the effect of the fuel side. Then you come down for the 3g throttle, and throttle back up. No problem. Typically when we transfer propellants, we get a little hotter propellants (talking about the upward curve between 150 and 200 seconds). In a 300 second or less duration test, we can go without transferring propellants, it just comes out of the tanks that sit above the engine. Over 300 seconds, we have to transfer from barges that are at down on the water. During a transfer, the propellant heats up 1 or 2 degrees, on the fuel, but you definitely see the effect. The lox heats up too. The engine has to power up because it has a lower density fuel. The fuel expands as it is heated, and you need more volumetric fuel to get the same amount of mass fuel. You'll also see venting effects. You look for little bumps, less than 1% is no biggie. The bigger effect is venting on the lox side (lox is more dense). "Here" you see the same thing, power up, power down, effects of the vent and repress. Here's the throttle up, a little more than I'd like to see. You started to repressurize the system and bring more pressure in up "here". You require the same amount of discharge pressure "here" but are giving it more pressure "here" so you're asking the pump to work less hard. As a result you see this power down here (downward slope on opov command at 150 to 180 seconds). It is very important for the system to recognize the effect of the inlet conditions. Don't forget that when we showed the lox inlet conditions, there was a real (big) jump right off the bat when you were repressing. You're all slowly reacting, then

takes a big jump at 150 seconds (looking lox inlet condition plot, can't tell which). Now, if we didn't have a vent on this test, we'd call this an anomaly. (looking at command limit trace now) We set the (opov) command limit during start and that's what you see here. If you had some kind of perturbation in the lox system that would cause the opov to run up, it would only allow you to power up until the opov reaches the command limit. If you have to drop off pc, whatever, you can't run past this point. If you hit this limit, there's something wrong with the engine. If something's REALLY wrong, we still have the redline on the lox turbine temps, and will shutdown the engine if the turbine temps reach that redline. This is still a safe mode (when the opov command limit is reached). The command limit is set during the start transient, and is basically based on the pump efficiency. It modulates the limit with pc. Note that it does not modulate it with engine inlet conditions.

PLOT 10

Here are my two fuel turbine temps. This is one of maybe ten parameters on the engine that REALLY indicates how the engine is operating. 20 seconds into the test, we're still stabilizing out. A lot of times, these channels will switch when you power down. You'll also see big spreads in these channels. I've seen spreads up to 200-300 degrees. The 300 degree spread was unusual, that was on 0215, with ccv changes and everything else. They will switch sometimes down near the bucket. Usually, the B channel runs above the A channel. It's due to fuel dumping out and hitting one sensor more than it hits the other. This (230-280 seconds) is normal on the 3g throttle. You're just hitting a different efficiency. We'll see this more on the low pressure fuel pump. So, we're looking for a nice, smooth trace, making sure one doesn't jump without the other one. If one jumps and the other didn't, we look at the fpov, fuel preburner press, pump speed, and if nothing else changes we say the coolant flow redistributed. You'll hear a lot of that. The power req's didn't change, and in reality, the turbine temps didn't change.

PLOT 11

Here are the lox turbine temps. Note the greater response to venting and repressurization (denser lox gives greater response). We typically get the overshoot on the lox. See it start to rise? (60 to 150 sec's) This is where you start to vent the lox inlet. Same press here, lower press here, pump has to power up. Looking for spikes, shifts, that type of thing, and absolute levels as well.

PLOT 12

HPFP disch press. This will be very flat because we're trying to run a certain amt of fuel thru the engine. This won't be as flat as HPOP disch press. because the only thing downstream of this (HPOP) is the injector that's a constant resistance, and a pressure here with a constant amt of lox flow that's coming thru. This guy (HPFP) has a whole engine downstream of him. You may see a little variation (10-30 psi). If you see big shifts, look for instr., and other problems. This one has a CADS backup. We trust this one more right now because it has better fidelity, but it fails more often. Facility PID is less coarse. In general, facility pids fail more often than CADS pids. CADS has bit toggle - cannot get anything less than that. This is a problem with CADS.

PLOT 13

HPOP discharge pressure - nice and flat. Overshoot is expected. Has a CADS backup.

PLOT 14

Here we start getting into test comparisons. Chose 554 because it was the same eigne and same thrust profile. Different pumps but this gives us a good comparison. Same greenrun profile. Comparison test gives you an eyeball indicator.

PLOT 15

This tells us what the vent is doing - two tests! Very repeatable.

Block I controller has a much bigger bit toggle than block two.

PLOT 16

HX discharge pressure. It is very unusual that you get such a good comparison test in terms of power level transients. Looking for shifts - nothing unusual. This takes a long time to stabilize.

PLOT 17

Fuel pressurization inlet pressure. This is the same thing but it's on the fuel repressurization line. Looking for any shifts we don't understand. This one will shift drastically if you do a repress flow change. If you change from max to min, you'll see this go way up. The previous parameter will change drastically in response to a max to min change in the gox flow. This is because you're changing the downstream resistance by a bunch.

PLOT 18

FPOV actuator position. Why the big delta? 2% is probably more than 1.5 sigma. It's because you changed the hardware. Have different pump and turbine efficiencies. A 5% change would probably cause some alarm because you probably couldn't get two pumps that were that different. Analyst will verify that other things such as preburner pc and turbine temps are also high - to verify that the pump is less efficient. Also, the fact that the difference is not constant from one power level to the next means that the efficiency delta is a function of power level. The pumps are more similar at 65% than at 109%. But also remember the valve window issue. A lot of difference at a larger opening only represents a small change in resistance. So even if the efficiency shift were the same at all power levels, you'd expect a larger spread at larger power levels.

PLOT 19

Fuel turbine temps. These are hotter - correspond to FPOV more open. B and A channels for both tests. The deltas are actually a little less between these tests than would normally be expected.

PLOT 20

FPC pc. It's a little up - this makes sense based on what we've just seen. This parameter will drift on some tests by 100 or 150 psi. This guy is mounted very close to the engine and is being thermally effected by the engine. The transducer is chilling down. We've put it on an isolation block - a barrier between the engine mount and the transducer (insulator). We still see a tiny drift - but not nearly as bad as before. When you see a drift here, ask two questions : 1. is it real? and 2. don't I have an isolation block on this engine? The isolation block is easy to figure out because you know the eigne or you call SSC. Verifying is it real is also pretty easy. If you see a 100 psi drift, that's a pretty good power change and you'd expect to see it in the fuel turbine temps, the FPOV, fuel pump speed, etc.

PLOT 21

No speed. Remember that from start? There is a CADs pid if you want to check it.

PLOT 22

HPFP discharge pressure. Very similar to previous test. No problems.

PLOT 23

Balance cavity pressure. We saw shifts on this test - it was the only anomaly we noted on this test. Some hashiness. Behind the third rotor of the HPFP is the balance cavity. We leak some of the discharge flow back there and it's used to center the rotor. There's a pressure measurement in there. We look at the delta-p between this parameter and the discharge to see if there's movement in the rotor. Pressure is wavering around (see at about 70 sec). This parameter is driven by two things - the fuel pump discharge pressure and the rotor position. There's nothing here that indicates a bad sensor. There's also no backup to this, so for now you believe it's real. If there's a doubt look at pre-start or post-test when there's no fuel in there, then you think you have a data problem. If there were this type of waviness in the fuel pump discharge pressure, you'd no longer suspect the rotor but now reason about the fuel pump discharge pressure.

PLOT 24

MCC coolant discharge pressure. The fuel comes out of the fuel pump and splits (at the diffuser). Part of it goes throughout the CCV, part of it goes down the nozzle and then joins the part that goes through the CCV and goes to the preburners. The other leg leaves the MFV and goes to cool the chamber jacket. There's a tremendous pressure loss in the jacket. You might ask why it's higher on this test. The high pressure pumps on this test are both different as is the LPFP.

PLOT 25

Coolant discharge temp. There's very little reaction to power level changes (notice the small scale). Looking for absolute levels here. This is a primary indicator of whether or not we're cracking the mcc. As we put time on the mcc, we'll form cracks and some of this flow will be dumped into the hotgas area. Coolant jacket is a piece with about 1036 slots cut in it - run the fuel through there. With a crack, you open up the fuel to the hotgas wall, it pours out and adds to the boundary layer flow and you see this temperature drop off. Primary indicator of chamber health.

PLOT 26

LPFT inlet pressure. This is just downstream of MCC coolant discharge. Recall that MCC discharge pressure was a little high, this is a little high - makes sense - just have a more blocked turbine. Would expect from this that the LPFP speed would be up.

Plot 27

LPFP speed - it's up, as expected. You're driving the pump speed up with higher inlet pressures. These are the kinds of things you go through to confirm one parameter with another. This can eliminate a lot of what could otherwise be considered 'anomalies'. If pressure weren't up and speed was, that would be considered an anomaly and you'd have to resolve it.

You're looking for a nice flat trace. We're looking for two things that typically give us a change in the pump speed - the first is a repress flow change. When you change from repress max to min you drastically change the pressure downstream of the turbine - change back-pressure. The turbine speed is being driven by the delta-p across the turbine. When you change the repress flow you change the turbine discharge pressure and you change the pump speed by 100-150 rpm (it drops). This must be recognized as a normal reaction to a change in the repress flow.

There's one other thing - piston ring seal shifts. This is a previously identified anomaly. In the secondary cavity there's a piston ring seal which has been known to shift. The LPFT flow goes out of the pump, does a lot of cooling - around the preburner, down the walls of the HGM (HGM is double-walled and the fuel goes through there) and then it dumps into the secondary cavity. From

there it goes up into the hotgas end and down into the main chamber. Separating the hotgas area and the secondary area and the pc area are piston ring seals. These are seals around the outside and have been known to shift in the past. When they shift they cause a change in resistance which feeds all the way back to the LPFT. When seal shifts, a lot more flow can pour down on hot wall. This will change the speed on the LPFP by about 100 rpm. We've seen one monster shift one Engine 2032 (or 2034) - 12 sigma shift. Had never seen anything like it. Have seen 100-200 rpm shifts due to the piston ring shift. The only way to recognize this is that you see the downstream pressure change (fuel press interface pressure) and you don't have a repress flow.

****DANGER ZONE****

You don't see a pressure upstream change, because this pressure wave doesn't feed to a great degree through this low pressure fuel turbine. You see a speed change that's reacting to this pressure change, and you say, okay system made self check. It's the only thing downstream that is not metal to metal hard. It is actually steel that is stuck in there.

Q: If that gives you fuel leaking into the MCC also do you see a drop in temperature there?

A: On the MCC temps normally you don't, but on this monster shift I was talking about, you saw a big change on the temperature. That's why we knew it was a monster shift, because we saw a huge change in the low pressure fuel pump speed and it actually showed in the fuel pump discharge condition. Not only did it change temperature but it changed the fuel pump discharge pressure - to feed all the way back here was a whopper. The vast majority of pressure drop was right in here.

PLOT 28

Okay here's your high pump inlet pressure, this is your discharge from your low pressure fuel pump and it is primarily driven by the low pressure pump speed and the engine inlet pressure. You can see the small scale. Again what you are looking for are reactions to the pump speed changes and in the scenarios that I just gave you of the piston ring seal shift - you'll see that in this parameter too. Because you raise that speed or lower speed by 100 rpm depending on how you are shifting the seal, you'll see it in here too.

This is the pressure that goes into the density calculation for the fuel flowmeter.

PLOT 29

Here is the temperature that goes along with it. Let me back up just one. Remember I was telling you on the three g throttle, you go through this dip in efficiency island and you see it in the low pressure fuel pump, that's that guy right there. All you are doing power level wise is coming right down, but you see it come back up and curve around and come back down, you're going through a different efficiency, I think it's on the big fuel pump and you see this reaction here, it's just the system balance to it. You see the same thing in temperature. This is the temperature, same location here - goes into the density calculation. Notice the very small difference in temperature - small scale here, but you do see the reaction to all the difference power levels. You're looking for the same type of thing.

You have seen the OPOV actuator position, again look at how close it is compared to the comparison test, even though you have two different pumps. We were amazed by that. Usually there is some kind of change. This was an anomaly on a previous test, see how it comes back up here, we look at that as an anomaly on the previous test - that's the kind of stuff you'll be looking at.

You go through the data until you see something strange. Well did my lox inlet pressure change to give me that, no. What else

could do it, did power level change, no, didn't to that. Then you start looking at the pump, did the pump turbine power level requirement change, I don't know, lets go look at the preburner pc. That is the kind of thought process you go through when you see something like this. You really just make a mental note of it and put it down on you sheet. We have a master observation list. It's a real fancy term for just writing down on paper what you see. As soon as you write it down you know then that you're looking for it in lox turbine temps and stuff like that.

PLOT 31

Here are the lox turbine temps. Looking more for comparisons because you've already seen this PID by itself. Already identified humps and jumps you don't understand - looking for the general trend. No problems here.

PLOT 32

Lox preburner pc. This one we don't have on flight, all we've got is this one facility pid, we don't have cad pid. This has been known to so something called icing. We have that in some other parameters. You can see again, the lox side overshoots a little on the start, you see as it's coming down as it thermally stabilizes you see the change in the power level, that kind of thing, you see a very slight change due to the lox inlet condition, remember you're venting and repressing. You can barely see it here, but it takes just a little bit of pc (along with the turbine temps) to give you the power change to overcome that inlet condition.

TAPE 4, SIDE 1

A: Let me tell you about icing. Icing, we have a preburner, here's your turbine down here, here's your dome you got lox and fuel pouring in here. You've got a sensor that is coming out here to the sense line and the transducer sitting over here. What happens is sometimes you actually, don't forget the product of combustion on this thing is basically steam, you got hydrogen and oxygen from some water some steam whatever, you'll get a little bit of steam that will build up in here and you actually form a water block and you'll form an ice block right in here, when you do that here what the transducer is reading now is only this pressure that is trapped in here, that's called icing, when you form that block in there. The typical trace you see power level transient, you see going like this whatever, and in that there will be some amount of hash mostly - with icing you'll see it go flat. You won't see any kind of hash nothing like that, you'll see the power level change on other parameters and this thing will say flat. Everything out here in the engine is changing, it's just you've got this block of ice in here that is trapping the pressure in there. Obviously, it's not real we have all kinds of other things that tell us that the preburner pressure has to be changing, but this guy is iced up. That's called icing.

One way to check for (icing) is post shutdown, sometimes in the test you'll see the thing come down and all that happens is that this is melted out and you form a little hole in there and allow it to come out and it all melts out and you see it come down. We are seeing a lot of that especially on the lox preburner.

PLOT 33

Here is the HPOP speed, the first thing you notice is you don't have the data for this test. Let's take a look at the last test which had the same profile. You see it react to the venting and the repressurization of the inlet. It's coming up again, reaction here, as you lower the inlet pressure the lox pump has to work harder you see the whole lox system power up, you see the pump speed, let me tell you about this, this is the only parameter that is a cads pid that we don't get from the cads, pid 2, I don't know how they assigned it, they just came out with it, we can actually get this data from the dynamics guys. They take their accelerometer data and translate it over to a pump speed and they

ship it over to our system and we merge it in. We don't actually measure this pump speed, we don't have a pump speed measurement on the lox pump. We have it on all the others, but not the lox pump. Typically we don't get this data right off the bat, 9 times out of 10 this plot doesn't have the current test data entered, because they plot these things overnight and by that time the pump speed hasn't been merged in there.

Q: Does this pid typically help you resolve things?

A: Sometimes it does. If you have a lox side anomaly during mainstage (that's important) then yeah, it's very useful. During start and shutdown as it stands right now you can't use this pid because it steps in 2 thousand rpm increments. It's just the ability they have to process the pid on the transient. For all practical purposes we don't have lox speed during startup or shutdown. What we have used are things like preburner discharge pressure to give us an indication of what the lox pump speed is doing.

PLOT 34

The HPOP discharge pressure. If there is a difference, you'd better look for a lox flow difference. Remember what I told you about this, two things determine it, lox flow and main injector resistance. If you assume the main injector resistance hasn't changed (which it shouldn't because there's nothing in here that should change). If you see a difference here, you expect to see a lox flow change. You see it's all very stable, right on top of each other, this is primarily determined by the engine. This is determined by the injector resistance which is part of the engine.

PLOT 35

Here is the lox injection pressure, same type of thing except it is slightly downstream and actually down here in the lox dome. This area is called the hot dog. It looks like a hotdog. (FYI: The heat exchanger bifurcation joint. The line going into the heat exchanger within the exchanger actually breaks off into two lines, called bifurcation joint or "baby pants".)

PLOT 36

Here is lox injection temperature, there are two tests (on this plot) and one temperature. It's in the lox pump. If you could see this, you could see right off the bat, hey I got a hairy injection temp here and it looks like a piece of instrumentation has gone bad. We know this because we know from previous experience what we should be looking at, in fact we almost see the other test. It looks like it's the previous test that has failed, but what you want to do is plot that individually to make sure which test it is.

PLOT 37

Preburner Pump pressure, you know what it is, and the only thing you can say about it, you are looking again for a comparison with the other one in terms of absolute level and you are looking for the reaction of the vent and repress to make sure nothing else is going on. Looks fine. Now when I say the only thing, there is another part, remember I told you that this is the first part of the job, sitting down and analyzing data. On this test, just as an example, there were objectives on the test to evaluate three out of four pumps for greenrun, there are certain requirements that we have to greenrun and they are all spelled out in the specs. This is one of them, there has to be a certain amount of pressure from the preburner pump at a certain power level and a certain lox inlet pressure to pass that pump for flight. We have programs that do all that, that check all that and make sure we don't have any problems. If you have too low of pressure you're going to adversely affect the system, as we said before the valve may run too high, you may run out of room, the valve may be running at full open and you can't control the engine anymore the way you want to.

PLOT 38

PBP Discharge temp. If you see a small deviation, you wonder what it could be driven by. In our mind, we already know because we have probably dealt with a problem like that before. Is it driven by the preburner pump or engine inlet? If I feed hotter lox to the engine inlet it's going to give me hotter lox all throughout the engine and we'll see it here as well. So in this case you would reason, is the preburner pump driving that - is it running a little harder than it should and therefore giving me a higher temperature, have I changed my parasitic flows so that at a given speed and a given engine inlet condition I've got a little hotter lox coming out of the preburner pump, all those kind of things run through your mind, just go through scenarios and stack them up and start running through the data so that you can to eliminate scenarios.

PLOT 39

Here is heat exchanger interface pressure. This is just discharge from the heat exchanger. You've seen the data before and now we're looking at it just with another one. This is the one you'll see change drastically if you do a repress change: you'll see it shoot up or down depending on which way you're going. On this one we were running max repress through the whole thing so if you saw a drastic shoot up here, let say 150 seconds, first thing you would think is did we do a repress change here. If you did and the gain turned out right - you've nailed it.

PLOT 40

And finally in this box, is the heat exchanger interface temp. Same kind of thing, see how long it takes to stabilize out here. Remember you have an initially cold system, you've got lox turbine temps on the outside - long time to stabilize the whole system out. You see the effect of the venting/repressurization because it affects the lox turbine temps that are on the outside of the heat exchanger. That's what's heating the lox that is in there.

PLOT 41

Pogo Precharge pressure - very repeatable, no unusual shifts anything like that.

PLOT 42

Again I'm ... not spending a lot of time on this - you're looking for the same kind of thing. You're looking for levels, reactions to vents, etc. You're seeing the lox system reacting much more than the fuel system does to the engine inlet conditions, which is what you expect.

PLOT 43

Here is your HPOP inlet pressure, remember how your HPOP discharge pressure was nice and flat through the whole venting and repressuring and everything else, all this is because you're changing the conditions here, you're venting initially then pressurizing and venting again and all it is feeding right through here. ... You actually have two channels on this and you'll see that as we go through the instrumentation package.

PLOT 44

Last one in this package, here's the MCC hot gas injection pressure. This is the only pressure measurement of the hot gas downstream of those turbines. This is actually a measurement that's in here on the lox side of the main injector. - right in here measuring gas pressure down here, you would see a shift in pc and very drastic shifts in turbine efficiency and that type of thing.

This one is known to go bad quite a bit. It a very nasty environment in there that it's trying to measure. This is also one that will ice on occasion. Remember talking about icing on the lox preburner? Anytime when you're producing the products of combustion you could run into icing.

GRPUMPS Package

Here is GR pumps, again it's the key words, don't worry about it, says high pressure fuel pump, it's actually both pumps. Now the component guys, Glen and that gang, they're machinery guys, are most interested in this package, we also look at it, we are familiar with what it says and we go through because you got to understand the and deal with the interactions between the system and the pumps.

CALC 1

Here's the pumps discharge temp minus the inlet temp. This gives an idea of a couple of things. One is the level of efficiency of the pump. This one is running about 57 degrees at 104, it's got a nominal pump maybe a little over. You don't see any kind of shift that we don't expect from the power level changes. If you did see a shift, that would mean that the rotor is maybe moving to a new position that would indicate a massive shift in the rotor. ... But it's not enough to give us any kind of significant shift on the delta p, which means it's not massive enough to give us a change in the parasitic flows.

CALC 2

Here's ... the Delta P, pump discharge pressure minus the balance cavity pressure. Remember that fuel comes in here, goes to the first impeller, second impeller, third impeller. The whole time it being pumped up. You're increasing pressure, it comes out here the majority goes down this duct right here. Little bit goes down here past an orifice and goes right behind that impeller. In other words it comes out the front of the impeller and sneaks around the back of it and runs down the back of it, and that flow cools this whole turbine area. We measure the pressure back behind that impeller, we measure the pressure in front and the difference gives us an idea of what the rotor is doing. That's what you see here. You can see a shift in the power level, looks like a little problem here, shift down okay here, we come back up and looks like we are really moving the rotor around. You can see in here, in fact, right in here (approx. 170 secs.) somewhere we have a shift of 109 we don't even see. Don't forget we power up to 109 somewhere, you don't even see that in here, so there is a massive indication that the rotor hung up.

In fact ... the throttle transient starts at 220 seconds it's in here. We start throttling down right about here, it just about here that the rotor recognizes it. When you're analyzing these things one second is worth the difference of this happening and that happening is an infinity. We very often look at, "hey is everything happening at this data point?" In other words, shift here to this shift, does this shift, does this shift. A second is an infinity. This is ten seconds worth. If things happen two data points later, we account for that in our analysis as well. So there is a massive indication there you have problem with the rotor position.

CALC 32

The coolant liner pressure and the upper red line, I won't go to much into this one, because this is not, the system doesn't concern itself with this. This feeds off the system but the system doesn't feed off this. This is just the pressure down here that gives you an indication of the coolant going down to the turbine. Leave it at that, because again if you do a system analysis you won't be overly concerned with what it has to say.

Plot 4

Coolant liner temp - same type of thing. It's just there to measure the temperature. That's more of a turbomachinery type of thing.

Plot 5

Here is fuel pump drain pressure and whatnot. Now a lot of these in doing a system analysis you're not going to be concerned with at all. Do you want me to go through them or skip over them?

Q: What do you do when you go through them?

A: We go through them and we analyze them mostly for the turbo pump people. In other words, they look through, we look through them it like a double check really. But in terms of if we have to do a system analysis, what I think is being asked right now, we won't look at this.

Q: When you look at them, are you doing a system analysis of the pump or are you doing a turbomachinery analysis?

A: If we see massive changes in the pump performance, like the balance cavity pressure, then it is a pretty good size anomaly. We will probably look in these, and the reason being this gives you an indication of the parasitic flow discharge. Everybody knows what parasitic flows means, when I say parasitics?

Parasitics - anything in the engines that comes into the engine and basically does not go into the main chamber is a parasitics - it a parasite. It doesn't produce thrust. The repress flow is parasitic: does not produce thrust, it is necessary, we need it and it serves a useful function, but it doesn't produce thrust so by definition it's considered a parasitic.

A lot of it (parasitics) is leakage passed the seals, leakage to cool turbine, stuff like that. In general it's anything that comes through the pump that doesn't go out of the pump that is used and goes past seals and that type of thing. It will dump out the fuel drains. This is measuring the pressure in that drain so if we see a massive shift or whatever at least in the system's data, it looks like it's being caused by the pump and we theorize we may have lost the seal in the pump, we'll probably go straight to this guy. Because if we lost the seal on something like that you would expect to see massive reaction here, because this is where it's all coming out. So in that case you would use it in a system analysis, but really you break it down to "hey, we have an anomaly in the system", we take the data, we get over here and say it's a fuel pump anomaly and this is just one step further, saying it's a seal anomaly in the fuel pump and this confirms it or doesn't confirm it, one or the other. That's how a piece of data like this is used in the system analysis.

Plot 6

In the drain temps - same type of thing, but it's the temperature instead of pressure in the drain line. If you have massive drain failure, something like that it would go straight through the floor. It would be dumping liquid hydrogen out of it.

Plot 7

Here's your balance cavity pressure on the HPOP. Pam's module is doing an analysis on the HPOP balance cavity and quickly tell you what it is doing. It's looking at these two, but the fuel pump has fuel basically coming in one end, running through and coming out the other end. The HPOP has lox coming in, it wraps around comes in - it has an inducer on both sides - and it comes out through the middle. The way that you balance this guy out is you have a balance cavity up here and up here and basically what it

uses is the pressure the pump creates to balance the rotor. All it is, is two different cavities on either side you pressurize them and depressurize them to keep the rotor balanced in the middle. That is what you are seeing here, the pressure on the two sides. You're looking really for two things.

First, are they nice and flat if you see them both of them shifting in opposite direction here, mean the rotor has moved over for one reason or another then it comes down. The first question is: is the system driving it or is it the pump driving it. Do we have a pump problem or is the pump reacting to something the system is doing to it. Typically what you do is ... then start working your way (through). Is the OPOV open up, did we see the preburner pressure change, that type of thing. That's when you get down to the nitty gritty of each data point.

Second thing you're looking for is leakage, where one of these will move and the other won't, it's not real rotor motion. It's just that one of the sense lines is leaking. Again in a system analysis sense, really what you're looking for is if you see a shift you want to know if it's driven by the pump or the system, so you know which part to go out there and pay special attention to in the hardware inspection.

Plot 8

Intermediate seal pressure - this is the pressure that is the buffer between the pump end and the turbine end in the HPOP. This is the one we will always purge, no matter what. This is the only one that is purged full time, pretest, post-shutdown, everything, no matter what.

You see it coming up here, the reason it's coming up is that the seal is actually closing down as the rotor grows. As the seal closes down you get smaller clearance and the pressure that is upstream rises and you can see the effect throttle comes up here still closing down, notice out here at 200 some odd seconds we're still closing down, it is still stabilizing. The system is not fully stabilized for at least 300 seconds. Plot 9

Here's the secondary cavity pressure. We can really get into this if you want... It's up to you guys. I would think in building a systems analysis module, this would be one of the last things you would look at. Do you want to go into it?

{decision made to skip rest of turbomachinery package}

Again in the system module, what we initially do is, "Is it the pump, or is it the system?" And then we would break it down, using this data, as to where in the pump it is, if we determine it's the pump.

GRINSTR (Instrumentation package)

This again would be in the mainstage, this is basically a check at this point, if you have followed this sequence through you have a real good idea of what is right and what wrong on test. This package is built primarily ... to check the instrumentation. We built the program in C that goes off and checks all the instrumentation plus others and it does a very good job, we almost don't need this package anymore but we are going to use for a good long time until we have full confidence in the program.

Plot 1

A bunch of this you'll see are dual temps. Here are the anti-flood valve temps downstream skin temps. You'll see a slight reaction to the throttle transients and that type of thing. Remember we have an anti-flood valve right here, prestart doesn't let your lox go down here to the heat exchanger and post shutdown which will get below about 140 psi, it will shut off. Its only reason for being (the anti-flood valve temps) is to make sure you don't have a leak when you're not supposed to be pumping lox

through there. Prestart, post shutdown, (and something unintelligible). What we're checking here is to make sure the instrumentation doesn't go south on us. We are not analyzing a thing with this, all we are doing is checking the instrumentation. If one of the debonds (debonding just means come off that type of thing) you'll see it go way up.

Plot 2

Main fuel valve downstream skin temps. Same type of thing except you're downstream of the main fuel valve on the line. Remember that these are actually taped on the outside of the duct. They are not measuring the liquid, they are measuring the duct temperature, the metal temperature. Again you just want to make sure that one of your pieces of instrumentation doesn't go south during a test, because the next chance you're going to get to see it is the beginning of the next test and you'll want to fix it way before you get it to chill on the next test.

Plot 3

Gox supply line temp. This is the one that's making sure that the bubble is still here in front of the OPOV - same thing.

I think we will tear through a good part of this package, because you're going to find that the main theme to this entire package is redundancy. We check redundancy. That all this thing is doing is checking multiple channels. OPOV, we've got an A and B ... and you're checking to see what is going on here, it's pretty close it's easily within a half of a percent, no problem.

One thing to say about the OPOV and FPOV A & B. Normally the controller looks and controls with the A channel, It's got two bridges coming off that one actuator. It normally uses the A channel. If the A channel shows a problem, the way the controller knows that, it knows what it's commanded position is and it can read the feedback position. If the command and the feedback get off of each other by 6 percent, the controller declares that channel is shot, and it will actually come over here and control on this B channel. If that continues to control within 10 percent on the B channel, then it's fine - it runs along and is as happy as can be. At this point when it throws out the A channel, it declares a FID (failure identification delimiter). It says, "hey, I'm going over to the B channel." If it fails the B channel by 10 percent it goes into hydraulic lockup, and all that is, it says, "Forget it, I can't control the valve anymore, I don't have control of the valve, it's 10 percent off of where I want it to be, lock up the hydraulics and the valve and let the thing run, and let's hope to God it stays together."

Q: When it changes channel you'll probably see a shift. (You do see a shift) How would you know you switched the channels?

A: The first indication is the fid, when we get that quick look, which is that one page fax sheet, remember it had that section on there, fids and comments. First thing you do is when you get those numbers, and you get a bunch of zeros and ones and sometimes you get the explanation, you look for the explanation, what it is exactly. Then you can home in and you know what it is at that point. If not, what you do is actually see these two channels flip. It's a weird thing to look at, but you'll see the two running like this, then you fail A, what it will do is clunk and it will flip. All you will see is this line in between, but all of a sudden your x's are running on the bottom and your triangles are running on the top. It's got to do with how the controller looks at the data. It always looks at it in this A position, so what it really does is switches the B into the A position and continues to look at the same position. You will see the system react to that. For instance, it says the two are half a percent apart right now, if that half a percent holds true, you'll see it switch by half a percent.

Plot 5

FPOV, again we're looking for redundancy, we have already looked at the levels, we've looked at trace, we've looked at how it's reacting and everything else and typically we're ... looking for redundancy.

Plot 6

CCV, the same type of thing. Now, this one we point out you get a little more spread then on the other ones. I have yet to figure out why, but you always get a little more spread on the CCV than you do on the other ones. In fact you see the spread get bigger as you throttle down, that's very normal. You didn't have a spread here and you do have a spread here so we are thinking that it is thermally biased.

The CCV is running on a schedule, remember I told you during main stage there's only two controlling valves the FPOV and OPOV. The main fuel valve and the main lox valve are full open all the way during mainstage. The CCV varies dependent on power level, that's it. If you're at a certain power level, then it's at a certain position. We've got that schedule if you want it.

It's not controlling inlet conditions, pc, or just anything else. It's just saying, "Where's my pc reference?" It reads it in and says, "OK, go to this position with your CCV."

Plot 7

There are dual redundant channels on your MOV, again the same thing, 6 percent and 10 percent, usually controlling on A. You see a little bit of change here, notice the small scale - no problem. It's running full open the whole time.

Plot 8

MFV is the next one, same type of thing: running full open; small difference here; half a percent of so; no problems. We've seen in the past, by the way, with these that come down and come back up, there's liquid air dripping on that actuator. That's something to note if it comes back.

Plot 9

This is probably the biggest one in checking the instrumentation. Remember that your OPOV controls to give the PC that you want. What you want to do is make sure you're reading PC correctly, because if you're not, the engine is going to power up and down to get whatever it thinks the PC is.

The way we do that is we have two PC sensors. Two transducers that are actually reading the main chamber pressure. They are approximately 180 degrees apart. Each sensor has a dual bridge, so we have four main chamber pressure readings. The pids 129, 130, 161, 162. So you've got two pressure transducers with dual channels readings each. One is identified as A and one identified as B. You'll see A1, A2, B1, B2. What they do is on the A channel they take an average of that and on the B channel they take an average of that. Then, they take an average of the two transducers and that is what you see when you pull up pid 163, which is pc average.

This is just taking the reference minus the A average. So it's looking at "this is where I want to be and this is what I'm reading." You're looking for spikes and that kind of thing. See all these? (reference spikes at 17, 27, 41, 52, 65, 167, 217 secs.) This is the effect of throttle transients. Here you're throttling up to 104 ... down to 65, 64, 63 ... 104 ... 109 ... and here's your throttle down. The reason these things happen is that when your engine is sitting at steady state, it's looking at pc reference and looking at pc on top of each other because it's making it there. So what happens is, all of a sudden, with pc ref., you go plunk and you drop it out and it takes the engine a certain amount of time to realize it and adjust to it. That's why you see all these individual steps.

So what you're looking for during the steady state portion is any kind of spikes. You're also looking for drifts; you're looking for difference from zero. Because difference from zero says one of your transducers is biased. In other words, if one of your transducers is biased by 20 psi, then what is going to happen is the engine is going to power down by 10 psi, because it's rating the average of the two.

Q: These spikes are always pretty slow?

A: That's how long it takes the engine to get down there and fully recover. It doesn't look on the pc plot, but that's how long it actually takes to get down there, and you notice that we're looking at a pretty small scale when you're looking at 3000 psi normally on pc.

Plot 10

Same thing except this is the B avg. Again, no differences here. If you see one of these as 20 psi high, you're going to see the other 20 psi low because it's averaging the two and using it.

Plot 11

Here's the A1 minus the A2. You can see very small difference because it's the same transducer. Even in the throttle transients, you see very small changes because you've got the dual sensor that's coming out of the one transducer - it better be the same.

Q: Do you ever see it reading different?

A: Yes. Sometimes you do because of the way you're processing the data, or maybe you're dropping liquid air on the sensor. It's unusual, but we've seen it in the past.

Plot 12

Here's the B1 to B2. Again, what you're looking for, especially we've seen ones where they've spiked and you'll spike by 30 psi and the engine will react to it. The engine doesn't know any better. It's trying to control the pc and it does it. It moves the OPOV around to try to get it. It doesn't know that it's spiking or anything else.

We do have disqualification limits on the sensor, I think we have a 75 psi between channels and a 400 psi overall limit and then you go into hydraulic lockup. But up to that point, and that is a big point, it's going to control to it.

With the plotting program that we currently have, you can only get a 1,000 points on a plot. That's very important because especially on this parameter on pc you'll have intermediate spiking you won't see a thing here and if you plot 20 second time slices, which gives you 1,000 points, you'll see it spiking all over the place. It's just that you can pick up only 1000 points with this program. Jean and Jeff have put together a program that works very well. It gives you every single data point.

Plot 13

Here's the MCC coolant discharge press. minus the LPFT inlet press. All you're doing is measuring from this point, the coolant discharge to turbine inlet. This is basically the delta p down this duct. That's it.

This coolant discharge press, pid 17 has a tendency to drift in test. Very thermally affected. You'll see this thing drift and in fact, in a lot of tests, you'll see this thing go negative. You know right there, that's not reality because if you have negative delta p, you'll have flow going the other way.

Plot 14

Here is the individual channel to the pc. You can see again

this is just another look at is there any spread, this should be very tight. If you've got a 10 psi spread you've got a problem.

Plot 16

All you see is the redundant channels. This is an average of the A and the B, 203 and 204, but cads pid (86) here is the average. The reason that you want to look at these is that this is what feeds into your fuel flow calculation.

Plot 17

Same thing on the temperature. Here's the temperature A and B and their average. No problems there.

Plot 18

Lot of the rest of it is redundant. The difference here is you got a cads pid and a facility pid. It kind of sort of redundant, but not as redundant as the other one. Your looking for the comparisons.

Remember we lost the HPFP speed on this test. That's going to be coming up real soon.

Plot 19

Here's the redundancy on the HPFP discharge press.

Plot 20/21

You're doing a double check on the levels and the shifts ... just to check the instrumentation. These are all the redundant channels you have in the system.

Plot 24

Okay, the hot gas pressure. Again you have two facility one and cads. These are actually measured in different places, so you see slightly different levels of pressure.

Plot 23

HPOP inlet, same story ... checking for redundancy.

Plot 24

LPOP speed, you'll frequently lose this parameter during flight, it doesn't hold together very well there is a speed nut that actually backs out and you loose that a lot.

Plot 25/26/27

HPOP discharge press/preburner pump - we'll skip these because again they don't tell us anything we don't already know. I'm looking for one in particular.

Plot 28

Lox dome temp and the ox injection temp: these are measured in approximately in the same location, we have always had this spread and we have never been able to figure out why. It's like a 2 - 2 1/2 degree spread. We tend to believe the facility. From studies we have done in the past, it tends to be more realistic.

Plot 29

Here's your fuel flow - This is interesting. This is the engine fuel flow. Both pids are engine fuel flowmeters, but one is as measured by the engine, that's pid 100, the other one is this guy as measured by the facility. Same flowmeter on the engine but one is an engine measurement and one is a facility measurement.

Plot 30

You've got your individual channels of fuel flow. In this case you only have one fuel flow discharge pressure. ... Let me skip a bunch of these because it's just the same story over and over.

Q: When you calibrate the engine fuel flowmeter, do you use this 722...or do you use pid 100?

A: You use pid 100. Well, you calibrate this flowmeter to what is up there. You don't use 722. You use 100 because you want the controller to be thinking the right thing. Usually, they're very close, but if there is a difference, you use the cads pid.

(skip plots 30 - 32)

Plot 33

Lox flowmeter - this guy here. Here is a flowmeter discharge pressure which is higher up in the line, this is a lox inlet pressure, see how it's spread out here and it's nice through the rest of it and we start throttling down. This is not throttling down but you see the effects of throttling down in here because you have a different size duct and different static effects at that point.

Q: That from the inconel, titanium switch?

A: No, that's from throttling down. This is the engine inlet pressure, this is actually from throttling down in two places, one is from the engine inlet and the other is up the line where you have the facility flowmeter. At that point you have a much bigger duct than you have here and you are measuring static pressure. All our pressures in the system are static. There is not one total pressure in the system. In measuring the static pressure you'll see a much bigger effect at the smaller duct diameter than at the bigger duct diameter. Because you have a bigger velocity change and that's what you are seeing there.

(skip plots 33 and 34)

Plot 35

You've got your vertical load cells. You've got three vertical load cells. We use them primarily in the system analysis when you have something that you think is affecting the chamber pressure. If we have chamber pressure anomaly, we want to know if it's real - go to the load cell and see if the thrust has changing. A simple way to check a chamber pressure anomaly.

More load cells, more load cells.

Q: Does everyone know what a load cell is?

A load cell is a giant overgrown spring gauge. It's what you use to measure thrust.

GRSHUT2 (Shutdown performance)

Shutdown-we have two packages in shutdown. First one is this one. The other is 2-sigma. I don't have that one, ... but it's the same type of stuff.

Plot 1

Cutoff times and all these times are referenced from cutoff on a test. The plotting system allow us to go ahead and line up different test on their cutoff. We frequently cutoff from RPL, which is 3,006 chamber pressure, we come down and ... just like start to look for repeatability.

We see our pressure come down nicely. No problems there. We

see most of the major parameters in the engine will come down in the first 6-8 seconds, something like that.

Plot 2

Here are our fuel turbine temps, and you can see even with the different fuel pumps coming down nicely, no problem; very repeatable.

One thing we look for is something called a cracked frisbee. A frisbee is a diffuser and a lox dome. The lox comes into the dome, it hit the diffuser and spread around evenly. The frisbee is hallow except for some ribs inside of it. We formed cracks on the bottom of the frisbee and the lox gets driven up into through those cracks and into the cavities that are in the frisbee between the ribs. So basically, you've got a lox storage area.

What happens is in post shutdown the pressure drops and the preburner allows the lox to escape and it dribbles out and you see this temperature going up. It will swing up like this, This one doesn't have a cracked frisbee, the ones that do, you'll see them come up much more quickly. In fact we have a criteria for cracked frisbee: from 5-10 second after shutdown, if the fuel turbine temp in either channel comes up 60 degrees a second or greater, we declare a cracked frisbee. You can go in with a boroscope and confirm it. ... We have a program that determines that too.

Q: Is this for both preburners?

A: No, Just for the fuel preburner. The lox preburner also has a frisbee, but it's much more sturdy, thick and it doesn't crack.

Q: What's the blip there (at about 3 secs.) ?

A: It's a typical characteristic of the way the engine shuts down.

Plot 3

This is the same thing for the B channel. What you can look for in these test also, if this temperature come way down here, down below 200, it's an indication that some kind of a fuel leak in the system. Recently, we saw several of those on a A1 test where the main fuel valve was stuck open.

The main fuel valve ... stuck open and allowed fuel to continue to come through there and we saw that one of the primary places you saw was these turbine temps which dropped below 200.

Plot 4/5

Lox turbine temps same kind of thing, you see it will come down no problem here. Again, if we're really looking for anomalies, we might analyze this (appr. 1-4 secs); I think this is when you go off in leakage on the OPOV, but I'm not sure, and all that means is that you've got one with a little more leakage or little less leakage that normal. But no problems.

Plot 6

High pressure fuel pumps speed and what we all notice about this is what we remembered from before is that it ain't there - it's down there. Again we have the other speed. Remember on this test just as an example, the post test hardware inspection showed it was very hard to turn this high pressure pump.

In that case, you would be very interested in "How did this thing power down?" If it was evident in the test, you would expect it to power down even sooner, maybe come in at 10-11 or 12 seconds before it shutdown. On this test I checked the CADS pid and you couldn't see it - it shutdown way out here. There was no indication in the data of the high torque, which means we may have caused the damage after the shutdown or well in to the shutdown.

Plot 7

The HPOP discharge pressure. Again no problem. Shutting down normally.

Plot 8

Fuel pump discharge press. Same kind of thing. Here's where (2-3 secs) where the pump actually goes into a stall region here. So it stays up and comes down here. Now the valves ... all coming down and shutting down in this region (no ref.) So all you're doing is spinning down the pumps and basically you're doing that with no fuel or lox going through the engine. The valve's already closed.

Plot 9

Here's your LPFP speed. Again, you see very repeatable. You see a little bit more energy in the system here (approx. .75 - 1.6 secs), we saw that in the previous plot, but we come down nice and repeatable.

Plot 10

Most of these have backup cads parameters which we already saw. Again, remember the speed was up a little (previous charts), a little more energy coming out of there (.75 - 1.6 secs.)

Plot 11

A lot of these, on the speeds, what we are looking for is called rotor grab, which is physically you grab the rotor with the bearing at shutdown you can see it drop down quickly here (ex., by 5 secs.), you don't see a problem here (on this plot).

On these, you're looking for comparisons of that sort. If you really want an explanation of what blips mean, I'll find someone to come up and give us an explanation.

Plot 12

Again, very repeatable; no problems.

If you get into a situation where you have a problem, we'll note it and we'll do the same kind of analysis we did on the mainstage, basically trying to track it down into other parts of the engine. In the shutdown phase, especially, it's much more difficult to do that because you don't have closed loop control on anything. When you have closed loop control you have a handle or tool that says "hey, I should have been doing this thing, and in fact I was doing that with the valves", or whatever. With this you don't have that, because a lot of this is open loop control.

Q: So from shutdown, from that point is everything strictly open loop control?

A: No. What you have is scheduled, and for instance ... The FPOV isn't allowed to close until of OPOV is 12 percent below it - those types of things. From there on you run along some fixed schedules. It's very repeatable in terms of valve positions.

The biggest driver on the differences in the valve positions in shutdown is where you start - where you come out of mainstage with the valves. One basic thing to remember on start/shutdown is you want to start fuel rich and you want shutdown fuel rich. What that means is that you start the fuel side first and you shutdown the lox side first. That keeps the engine cool, keeps the mixture ratio down and keeps the lox from eating things.

Plot 13

Skipped

Plot 14

The lox preburner pc; again you can see it come down nicely. This is where we turn on the preburner purges. Let's look at next plot.

Plot 15

Right around 2 sec (1.8 sec) we turn on the purges - see effect on preburner pc. This is the preburner shutdown purge we were talking about on prestart. On prestart we were purging these domes with pneumatics - we're doing that again to drive all that out of there. What we initially do we come along here and drive the pressure up, this is now reading helium pressure. We have a check valve in the system, it's not on this chart, it's on the PCA schematic but as it opens here, you see it opening and allowing the pressure to come down to the chamber pressure and at this point we basically choke and you can see it come along. This (triangle right around 14 sec.) is an effect of a post shutdown purge - on the POGO. Then you stop purging the preburner. You'll see the same type of thing on the lox preburner on your next chart.

PLOT 16

Here's the OPB shutdown purge pressure. You come along, you're running, you go with the pc and then you sit here purging, you take a little bit of that helium and you go and purge the pogo, you hit the pogo with the first charge and you turn the purges off in 15 seconds. This is at 13.5 seconds. Nothing post shutdown is done with nitrogen. Everything is done with helium. Reason being, on flight you don't carry any nitrogen, just helium. On a ground test, way after post shutdown you will switch over from helium to nitrogen on the intermediate seal. WAY after shutdown - helium is expensive, nitrogen is cheap. Everything in the first 20 - 30 second time frame is done with helium, nothing with nitrogen.

PLOT 17

Here's that POGO precharge. You've got a charge here (about 1.8 sec), you actually don't have the data out there but you have a charge out there. You can see come down here and you charge that thing, it keeps this ullage in here from collapsing. Again, prestart we did a charge, during start we did a charge, here on shutdown we do a charge. Then you come in here and follow the LPOP discharge on down.

PLOT 18

Anti-flood valve position. The anti-flood valve is down here in this line leading from the lpop discharge - the valve here is a pop-it valve that pops at about 140 psi and allows flows through here. This is just the opposite situation on shutdown, the hpop (should this be lpop) discharge gets down below 140 psi then closes. As a side point, we have had cases just recently where you will actually see this guy start to open again, reason being we throw firex (?) on the engine, firex is a system we have to put out fires, basically a nozzle spray system that is all around the engine. We have had some instances recently where we've detected fires post shutdown - you spray the engine and what happens is the lox that's lock up in this line (the MOV is closed, the two preburner valves are closed and the lox is locked up in this line) and when you hit it with that firex it warms it up instantly. It expands the lox and builds up the pressure and opens the anti-flood valve again. So we've seen cases on shutdown where this valve opens back up again.

PLOT 19

The high-pop intermediate pressure, again you can see going well out past the test and it keeps going. It is all being fed with helium.

PLOT 20

Here's the lox bleed valve, the fuel bleed valve...all the

ones we looked at from start enable to engine start. Saw the two bleed valves close and the POGO open - this is just the opposite.

We have plots that are typically used in a presentation to the chief engineer.

ERIK SANDER
Gains Analysis
5-1-92

- . Two parts of gains:
- . The controller (What is it doing in terms of controlling the engine.)
- . The physics of the engine. (If a pressure changes by x amount and has a certain resistances then pressure y will change by some amount. If this speed goes up I would expect this temperature to go up. This type of thing.)

THE CONTROLLER

- . The controller only has a couple of handles on how the engine works.
- . If you can understand these you can understand why things change in the engine.
- . Nothing magical about it. A lot of it is just flow, heat transfer and physics.
- . Understanding how the controller reacts to certain situations that are going on in the engine is really the bottom line of gains.
- . See Erik's book, Controller/Software.

Mainstage.

- . Normal operation. The controller is in control of the engine. Does this with 5 valves. 2 are full open during mainstage (MOV and MFV). Full open at all power levels in mainstage. The CCV varies with power level. (See plot of schedule). It says at this power level you will be at this CCV position. It is a function of power level.
- . Be careful. Testing with two CCV schedules at present. One is the Phase 2 schedule and the other is the Phase 2+. The 2+ is a new design on the powerhead which RKD has developed and we have started testing in the last year or so. A redesign of the powerhead which allows a more stable flow in the main injector area.
- . It does affect CCV schedule. For 2+ we must have a more closed schedule.
- . CADS (Control and Data Simulator) simulates a computer on board the orbiter. It sends Pc reference signal to the controller. The controller does not generate Pc reference signals. The CADS sends a signal that tells the controller at this time I want to be at this Pc level(main chamber). The controller will make the engine operate at that Pc level if it can.

Electrical Lockup.

- . A fixed orifice engine. We are not modulating the valves up and down to try to control anything.
- . If certain criteria are violated we will go into a phase called electrical lockup. The controller will send electrical signals to the valve actuator to hold the valves at a given position. Still using hydraulics.
- . The way the controller works is that it is controlling the engine with 2 preburner valves. The CCV does move when you are throttling but all it is looking at is Pc.
- . During electrical lockup the controller sends a signal that says using hydraulics hold that valve to that given position. For example, my Pc has deviated from my steady state value by some amount. If the controller is trying to control and Pc gets away from the reference point by 75 psi it says I have a big time problem. Electrically hold the valves where they are.

Hydraulic Lockup.

- . A fixed orifice engine.
- . Go into hydraulic lockup if we lose hydraulics. The actuator physically configures itself so the hydraulics are locked into the opening and closing pistons of the actuator. The valve is hydraulic locked up. It has cavities on either side of the pistons. The hydraulics are stuck in there and you can not move the valve.
- . The controller does not immediately know that the engine is in hydraulic lockup. It has no pressure access to the hydraulic leg. The controller thinks it is controlling the engine but in reality the actuator is in hydraulic lockup. There is a little valve in the actuator that when you take the pressure off of one side. It will physically shuttle over and lock the hydraulics into the actuator. The actuator is physically locked up but the controller doesn't know. It thinks it is still controlling just fine. What eventually happens is that the actuator will drift off because we don't have a perfect seal on hydraulics. When actual valve position gets 6% from the controlled position the controller throws a FID.
- . (TALKING AS THE CONTROLLER). I have a limit of 6% . I am monitoring the feedback from the valve positions (A and B channel) . The A channel says I am 6% off my command so therefore I don't trust it any more or I say I don't have command of the A channel any more. I switch to the B channel. I now open up my tolerance to 10%. I am now on the B channel and I say now if I am within 10% of my commanded position I still think I am in control. When the B channel gets over by 10% I give up. I'm out of control . I obviously can not control the valves and I declare hydraulic lockup. I throw the FID for the second channel that says I am in hydraulic lockup.
- . When it declares hydraulic lockup nothing changes in the engine because it has already lost control and just doesn't know it. We pulled the hydraulics 30-50 seconds earlier. It visually takes this long for the valves to drift out.
- . If you attempt a throttle transient right after you do hydraulic lockup it will know about it right then. Because you change the Pc reference command you will change the valve position command and you will very quickly get the delta between where you are and where the command is. Normally we see it because of the position drift off while the command is holding steady. But here we moved the command.
- . When you go into hydraulic lockup you shut the engine down with pneumatics (He). In hydraulic shutdown what you do is electrically controll the hydraulics going into the valves to give you these kinds of valve positions to shut down the engine.
- . Always shut down fuel rich to protect the engine.
- . The controller has no control once it goes into hydraulic lockup.
- . The engine has no hydraulics which is the arm between the controller and the engine.

The Controller Opening a Valve

- . The controller sends an electrical signal to the actuator that says Open!
- . That translates to the actuator as a little wand with a coil on either side. When it says Open! it energizes this side and pulls the wand over. The hydraulics are coming into that area and will change the way the hydraulics are going through different passages. That is what physically moves the valve. You make it go through passage B rather than passage A. If you don't have the hydraulics the wand moves over but there is no hydraulics going through so you have lost your control of the valve. During hydraulic lockup

its not important what the controller is trying to do because it doesn't have the ability to do it.

- . During electrical lock up it is different because we don't drift the valves because electrically you are holding at a given position.
- . More hydraulic lock-up test are done than electrical lock-up.
- . Electrical lock-up tests have been done and been shown that it can be done.
- . In hydraulic lock up we have problems with the valves drifting.

Normal Operation:

- . Two handles that the controller has on the engine. (OPOV and FPOV).

OPOV

- . Controls the PC.
- . Modulating it increase/decreases lox going into the lox preburner.
- . That increases/decreases the amount of turbine energy which is directly linked to the pump.
- . As the OPOV opens the pump speeds up.
- . Speeding up the pump drags in more lox.
- . The OPOV can control the MCC Pc by controlling the amount of Pressure that is driving lox into the chamber.

FPOV

- . Controls the fuel flow.
- . Fuel also goes into the chamber to provide pressure.
- . Controls the amount of oxygen going into the fuel preburner.
- . Controls the amount of energy that is available to the turbine that translates into the pump.
- . As the FPOV opens the pump speeds up.
- . More fuel flow is dragged past the flow meter.

Mixture Ratio:

- . Should be 6.011 at the engine inlet.
- . At all power levels we want this ratio.
- . Mixture ratio is the mass of the lox over the mass of the fuel.
- . Do not have a lox flow meter which the engine recognizes.
- . Each chamber injector is different. Each has its own efficiency. To get a certain amount of MCC pressure will require different amounts of lox and fuel. If the fuel flow can be set at some point in the engine where 6.011 times that amount of lox flow will give us the right chamber pressure then we have met all our conditions.
- . If a given amount of fuel flow can be set (for example at 104% PL 154 lbs of fuel flow) with the chamber and injector efficiency that is on a engine such that 6.011 times of lox flow combined with the fuel flow dumped into the chamber will give you the right amount of pressure then the conditions have been met. Condition of Pc will always be met. The OPOV is opening or closing to make that.
- . If the engine has full control it will always meet the fuel flow requirement and the Pc requirement.. Pc will equal to Pc reference and fuel flow will be determined by the C2 schedule(see Erik's book).

Example.

. My chamber is so efficient. I am going to set this amount of fuel flow. We run the engine and it says I need to set P_c so 5.9 times of lox comes in. We have met out condition for P_c but we have not met our mixture ratio which is now at 5.9. We crank down the fuel flow a little through C2. We still have the same chamber efficiency and the lox flow comes up a little. We now have P_c and a 6.011 mixture ratio. Bingo! The whole essence of getting the right mixture ration is setting the right amount of fuel flow. The lox flow is going to fall out to give you the right amount of chamber pressure.

How the Engine Works.

. We have a fuel flow meter in the engine. It is between the LPFP discharge and the HPFP inlet. It is very important to know it is in the duct, because when we change the duct we change the flowmeter. Each flowmeter has its own characteristics. This translates to us as K_f , which is a software constant. All it does is translate the engine flowmeter speed into volumetric flow. This is the most basic way of looking at K_f . If the duct has been changed then a new K_f must be used.

.. You have a preburner pump that is supplying a certain amount of pressure and flow. That pressure and flow is going to two places(the FPOV and OPOV). These two modulate off of each other. If you open up one valve a little it opens the other valve also. The controller is the only thing controlling these valves. If you open this valve then physically you get a little more flow and you drop this pressure just a little. Now the other preburner valve has a little less pressure. It's trying to hold the pump speed up but it has a little less pressure so it is driving less flow down into the preburner. The pump will slow down, the P_c will come down, and it will open up the OPOV to overcome it..

This is part of the Physics of gains. Keep in mind that there are other things going on in the engine that may overcome this.

. Now if you open up the FPOV you are charging up that preburner pressure. There is also a fuel split that comes down here. As you increase that pressure your increase this down stream resistance and it sends more fuel to the OPB. That would make you close the OPOV.

. The gains are not typically x drives y . It's $x + a + q + c$ all work together and they come out with some kind of a y .

. We have a pretty good understand of the individual gains. But we must marry these gains together. The only way we have to marry them is through the test data that we have experience with and with the model. The model knows the physics of the engine and it knows what the controller is doing. The short coming of using the model for gains is that until recently (since TTB data) there has not been data from internal instrumentation. We have had limited instrumentation and it has a lot of black holes.

The Facility.

. The facility has handles on the engine.

. P_c reference

. The throttle excursions we tell it to go through are set in the pretest.

. K_f and C2 set the mixture ratio.

. Engine inlet conditions

. We can to a degree control the temperature and pressure we deliver the liquid hydrogen and oxygen to the engine, especially the pressure.

. In the pretest we define a vent schedule. This is simply a schedule on a time basis that tells how we will feed the cryogen to the engine.

. We will start a test at a given pressure. We will usually vent the fuel side down and run the rest of the test at a lower pressure.

. On the lox side we typically start at about 80 psi. We will vent down to about 20 and then pressurize up to about 120-160 . We almost always shutdown at nominal(80 psi).

. If we do a fuel vent we rarely do a repressurization before shutdown. We can shut down at a lower fuel pressure.

. We can modulate the pressure. We don't have a very good handle on the temperature. We can set the temperature going into the engine at some given point. But we don't typically try in the run to increase or decrease the temperature.

. We do certain things that we know increase the temperature. The most prevalent one is the transfer. If the test is under 300 seconds we usually don't have a transfer. This means we have a fuel and lox tank above the engine and we run the engine off that fuel and lox in those tanks. They have been chilled before the test and we just drain them off. If the test is longer than 300 seconds we don't have enough lox and fuel to just run off the tanks. We can't physically hold enough for the test so we must do a transfer. At Stennis it means there are barges in the canal which are filled with lox and fuel. There is a line going up to the tank and we are transferring from those barges. The fuel and lox on the barge is cold. It goes up the line going to the stand it heats up by maybe 1 degree. As the fuel comes from the transfer line into the tank it dilutes . It dumps into the bottom of the tank which has a baffle arrangement. As it is mixed it slowly but surely heats up the fluids in the tank. The fluids coming into the engine are a little warmer. If you look at either the lox or fuel inlet temperature you can see exactly where the transfer starts. The temps will be nice and flat and then they will begin creeping up. This means higher temperature at a constant pressure which means lower density. More volume has to be pushed through to get the same mass flow. The system powers. Can see it in the FPOV and the OPOV. That is the kind of gains that we know are in the engine.

. Know before if there will be a transfer and start it at about 10 to 20 seconds.

. The system needs to know if there is a transfer so it can reason on it.

. The pretest tells us when the transfer starts.

. It is probably a manual input on the test stand.

. We had a recent anomaly on an A1 test where we were transferring and the temperature went through the roof. We drained the tank down to a certain level and we were getting impurities, nitrogen that is diffused in the tank. The temperature went sky high. This is something that will have to be recognized as an anomaly by the system.

Repress flows.

. Fuel repress flow.

. It goes out just down stream of the LPFT.

. Fuel comes into the engine through the LPFP out of the pump end, into the HPFP and is pumped up and comes down and then splits to different places. Part goes to cool the nozzle, part to cool the chamber, part ends up at the preburner from the nozzle.

. After it goes in and cools the chamber it comes out and goes into the LPFT and drives the turbine which spins the pump. Just downstream where it comes from the turbine it splits off and some goes to the fuel repress line.

. This fuel repress can be dumping at min repress(.2 lbs per sec) or max repress(1.2 lbs. per sec). We can change flowrate for any test by up to a pound a RPL condition. This has an affect on the components in that leg. In the LPFP speed you will see a shift of about 100-150 rpms. The gains are not very solid. Different components shift different amounts. We have ranges we have seen in the past. We can provide you with those ranges.

. As we look at the data we may see the LPFP shifted by 150 rpms. First thing we think, did we do a throttle transient. No, we didn't because we know what the pretest said and we seen the Pc reference. Look at engine inlet conditions., No, we probably can not change them that fast to give that kind of shift. Throw that one out. Did we change the repress flow at that point? Yes, we did change it. The shift is attributable to that so it is nominal. We should see a shift. If we don't we should flag it.

. The amount of change is important.

. In example above if there had not been a repress change then the shift in LPFP speed would be an anomaly.

. Lox repress flow.

. 1.1 lbs/sec min , 2.35 lbs/sec max at RPL.

. No repress flow or nominal.

. The three things above, Pc reference, engine inlet conditions and repress flows , are the only thing we do to the engine. Anything else is an anomaly.

THE PHYSICS OF THE ENGINE.

. This pressure should affect other pressures by this amount.

. If I see pressure go up and down is it flow related or resistance related.

. Two things that can make a pressure change.

. . The resistance in the line. Similar flow and increase downstream resistance that pressure goes up. Increase upstream resistance that pressure goes down.

. The flow in the line. Same resistance and increase flow to a point the pressure will go up.

. Fuel split Fuel coming out of the fuel pump splits to the main chamber and part cools the nozzle and eventually goes to the preburners.

. Pressure is determined by the resistance in the line and the downstream pressure.

. Resistance in the line does not change but the pressures do.

. The major physical gains are documented in the delta book by M. Alvarez and model gains by B. Piekarski(these are documented where they are good and not good).

. Important to understand where the gains came from and what kind of confidence you have in how the gains were built.

. In our heads we are updating gains as we go.

. Gains in powerlevel and turbine temperatures are pretty consistent.

. Delta p across the HPFP is affected by type of discharge duct.

Pressure went up, is it flow or resistance?

. The fact the flowmeter is going to run a given amount of flow is important.

. Flow is an anchor point.

. Resistance changes are actually hardware changes.

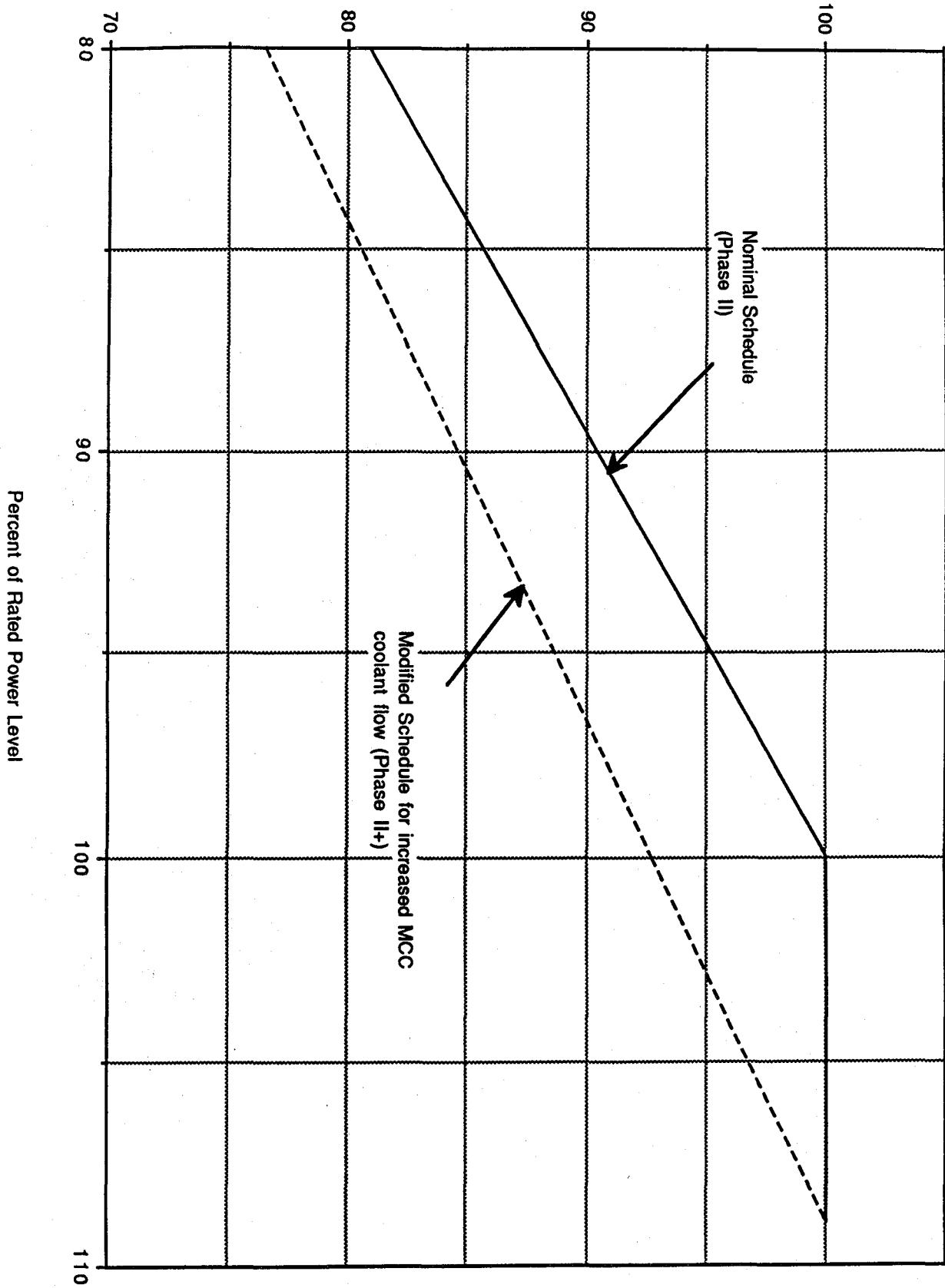
. Case study A1-531 is a good example.

Anchor Points.

- . HPOP discharge pressure
 - . Fixed by the amount of flow through main injector.
 - . Lox pump doesn't affect HPOP discharge pressure.
 - . If it changes there is a definite reason.
 - . If it doesn't change you can use it to look at other parameters.
 - . A point in the engine that few things feed into.
 - . Same engine, same flow and powerlevel you should have same HPOP discharge pressure.
- . HPFP discharge pressure.
 - . Less reliability than lox side.
 - . Has more things feedind into it.
 - . If HPFP goes down it could be a variety of things. For example, the MCC is leaking, the preburner pressures change, etc.
- . Main Injector resistance.
 - . Very good anchor point.
 - . The resistance doesn't change unless you have a hardware change.
 - . If you have ice it will plug things up and cause trouble. This is why we make sure engine is dry at prestart.
- . Lox Flow.
 - . Good anchor because it can be measured.
 - . Not so good because we see test to test variation.
 - . All the lox that comes in doesn't go down main injector leg. There are branches to the preburners and LPOP.
 - . If you see a shift there are few things that feed into it.
- . MCC Pc
 - . Pc will always be at Pc reference.
- . Looking for a one point failure. It is very unusually to get a 2 point failure at the same time.
- . Put higher priority on what the stronger anchor points tell you.

A3-109

CCV Position (% of full open)



1May92 Gains - E. Sander

OPOV is fed by everything in the engine, and each of those feeds has an individual gain. Some feed to other parts of engine. The lox inlet pressure feeds to the fuel flow split which feeds the OPOV. Each of those has a gain. Some gains are based on consistent pattern of engine operation. Others are not, the characteristic gains may be associated with a hardware change. This adds more gains to the picture. Cumulatively, the gains are not well defined. When you add, the greater your chance of inaccuracy. If you are going to use the gains approach, you need to find a very good anchor, such as the HPOP discharge pressure. This is much better than the OPOV. The goodness of the gains will also affect the ranking of the conclusions.

What else do we need to discuss?

Tim: Can we now look at what the controller is doing during start?

Erik: Start is easier to figure than mainstage because it's open-loop control. Follow valves, because they are what is changing. We don't effect repress flows or inlet conditions. You can't track P_c because it is changing, as is the fuel flow. For mainstage, these are two good anchors. The physics for start is more consistent, because there are less facts involved. Start is divided into three regions: 0-.74 with open-loop control, P_c control which runs .74 through mainstage, mixture ratio control that runs 2.3 seconds onward. 0-.74 sec has very repeatable valve sequences. OPOV low means P_c too high. Sigmas are smaller, because we run starts consistently, always starting to RPL. In mainstage, things start to diverge, like the turbine temps. During the early open-loop phase, the turbine temps will lie on top of one another. In mainstage, when you start controlling, the turbine efficiencies start to come into play and your turbine temps start to diverge. We try to control to a given point with turbines of different efficiencies. Your sigma spreads in mainstage, you have more uncertainty in the way the engine will operate. Some things settle out with the arrival of mainstage, such as main chamber pressure. This is because we are controlling to that function. In terms of anomaly resolution we would best illustrate with the CCV failure. CCV actuator was not coupled with the valve. While we were trying to control properly, and the data showed that we were controlling properly, there was a change in the engine. This is where the physics comes in. We were reading the actuator position, not the valve. The two were uncoupled.

In OPOV and FPOV, we had the CCV causing us to run abnormally. The controller was trying to modulate the two preburners to try to get back into nominal P_c .

We have alot more instances of gains associated with changes that help to explain mainstage than we have for start. Start is usually so consistent, that we don't have the data. It will be tough.

A3-111

I would try to work with mainstage first. Then try start and shutdown. Few people understand the start and shutdown transient. I would say that Dave Seymour understands start better than anyone around here.

RANDY HURT
4/30/92
CASE B1-143

This is a walk through of a ground test that has just run. Randy said this is data analysis that is still in the working.

START:

- . Interested because it was a ATD lox pump on a regular phase 2 engine.
- . Interested in the transient because an anomaly was predicted in the transient.
- . First 100% start on the ATD lox pump and ran at 100% for an extended amount of time.
- . HPOP discharge pressure surge was seen. Vanes are stalled until you get to a certain powerlevel. Once you get to that power level they become unstalled and engine powered up because it was stalled to begin with. Surge is up (Plot 1). Wiggle is usually not there. Apparent because the HPOP discharge pressure comes up. Actually lower surge than D. Seymour predicted with the Digital Transient Model. Not a lot of other data analysis to do about this except compare to DTM and present that.

MAINSTAGE:

- . Another analyst looked at the data before Randy and said there was an anomaly about 100 seconds where MCC PC actually went down.
- . Pc drop during mainstage is pretty bad. Supposed to be able to control it and keep it right at the reference power level.
- . (Plot 2) Comparison plot with another test. The Pc drops off and comes down at 100 seconds. Without the other comparison test the scale would be much bigger. It would look more drastic.
- . (Plot 3) Look at the valve positions of the OPOV and FPOV. FPOV has a wiggle in it also. At about 85 seconds the FPOV trends down a little and then turns around and trends up causing a delta of about 2%(a real eyebrow raiser).
- . This is a big performance change. about 2%.
- . From plot 3 looks like we definitely have a performance change in either the high pressure fuel or oxidizer pump.
- . Randy knows it was a development lox pump and he had seen this same wiggle happen on previous run so his first tendency was to think it was the lox pump.
- . Looked at level and it was a constant level with a wiggle that is on the border of what might be seen under regular engine control. But also we see it comes up to higher level. That's says we have a bigger energy requirement after this incident .
- . Probably have an efficiency change to the lox pump.
- . With that hunch the next large piece of data looked at was the HPOP speed.
- . HPOP speed drops at the same time of the wiggle and MCC pressured dropped. That is what drives MCC pressure so you would expect it to drop. The speed dropped of and then it came up to the same level.
- . If the speeds are at the same level and the pressure is at the same level then pump efficiency has not changed. If pump efficiency had changed it means more or less speed is required for the same discharge pressure.
- . Pump had a turbine efficiency change in the HPOT. This caused the valve difference.

. Going from 104% and back to 100% the valve position is up a little but its more is in line so the anomaly is almost gone away at the second 100% power level.

. Started looking at some of the pump internal instrumentation that is available because it is a Pratt Whitney pump.

. Nothing stood out and didn't see anything that said this is the problem. Didn't see any one parameter with large shifts. The turbine end showed more effects than the pump end.

. Went and talked to model performance person, Tracey Touchton, who had made plots for each of the development pump test.

. (Plot 4) This plot says the most about it (HPOT EFF vs TIME). The efficiency drops from 75% to about 74% at 100 seconds and stays at the lower efficiency.

. Know the anomaly did not cause a redline cutoff or anything like that.

. This plot (Plot 4) shows turbine efficiencies have dropped off and have come up a little bit. That pretty much gets the anomaly. The rest of the data analysis I will do is what it does do to the rest of the system. What this anomaly did to the fuel flow. Is it alright to test it again? A 1% change is not going to keep us from testing. I'm going to look at preburner pressure balance. This will give an indication of how the engine is balanced. I will look at a pressure balance and a pressure ratio. It will also help me indict the lox side I need to be able to quantify this.. A lot of what I have been thinking is intuitive.. This plot pretty much says it is the lox side. I must make sure it is not the fuel side.

TIM: How long did Pc dip down for? Did that come back up to normal again?

RANDY: Yes, it came back up again. It was down for something like 5 seconds.

TIM: Why did it take so long for the engine to adjust to this one sift? Is it the efficiency dropping off over time?

RANDY: Yes, that was why. From this plot (Plot _) it looks like (turbine efficiency) a step function.

TIM: If the efficiency had been a true step function. Say at one time frame and the efficiency dropped by 1% would Pc have adjusted almost immediately?

RANDY: For 1% like that it has to drop off and you have to see it in Pc first. So, Pc has to recognize it. There is a little delay there because of the system response time. The system responds fast.. The inertia of the pump is low. The valves open quickly and all that but you still have an over shoot or under shoot and it just takes a few cycles to get through that. A sudden increase like that it takes 2 seconds or something like that.

TIM: That is mainly time for the pump to spin up.

RANDY: For the pump to spin up and for the valves to open. All that is real quick so 2 seconds seems a little long. I would not get real excited if it took 2 seconds.

Right now as far as my data analysis goes I am in the middle of getting my presentation package together for tomorrow. I will look at these lox preburner ratios and there is a cross feed gain question that I want to ask someone about. If you open up the OPOV real quickly the FPOV responds. It opens before the system sees it. In other words if the OPOV opens then the FPOV open too.

TIM: Does that happen also during throttle up?

RANDY: During throttle up I'm sure it happens. Otherwise, you would get off mixture ratio. Your power requirements would go up. OPOV would open up and FPOV would lag or go way off mixture ratio.

The things I am going to look at like that are really just to rap it up. I am going to make extra charts, prime time charts, start configuration. But really the data analysis is mostly finished. I talked with the turbo machinery guys quite a bit and we have been throwing around hypothetical reasons for this efficiency change. We think we have a handle on it so we are waiting for the turbopump to come off to look at the hardware. We think it is either turbine bypass flow or tip seal clearance.

XX TEST 9040143

334

HPOP DS PR NFD 7K P

ΔΔ TEST 9040118

334

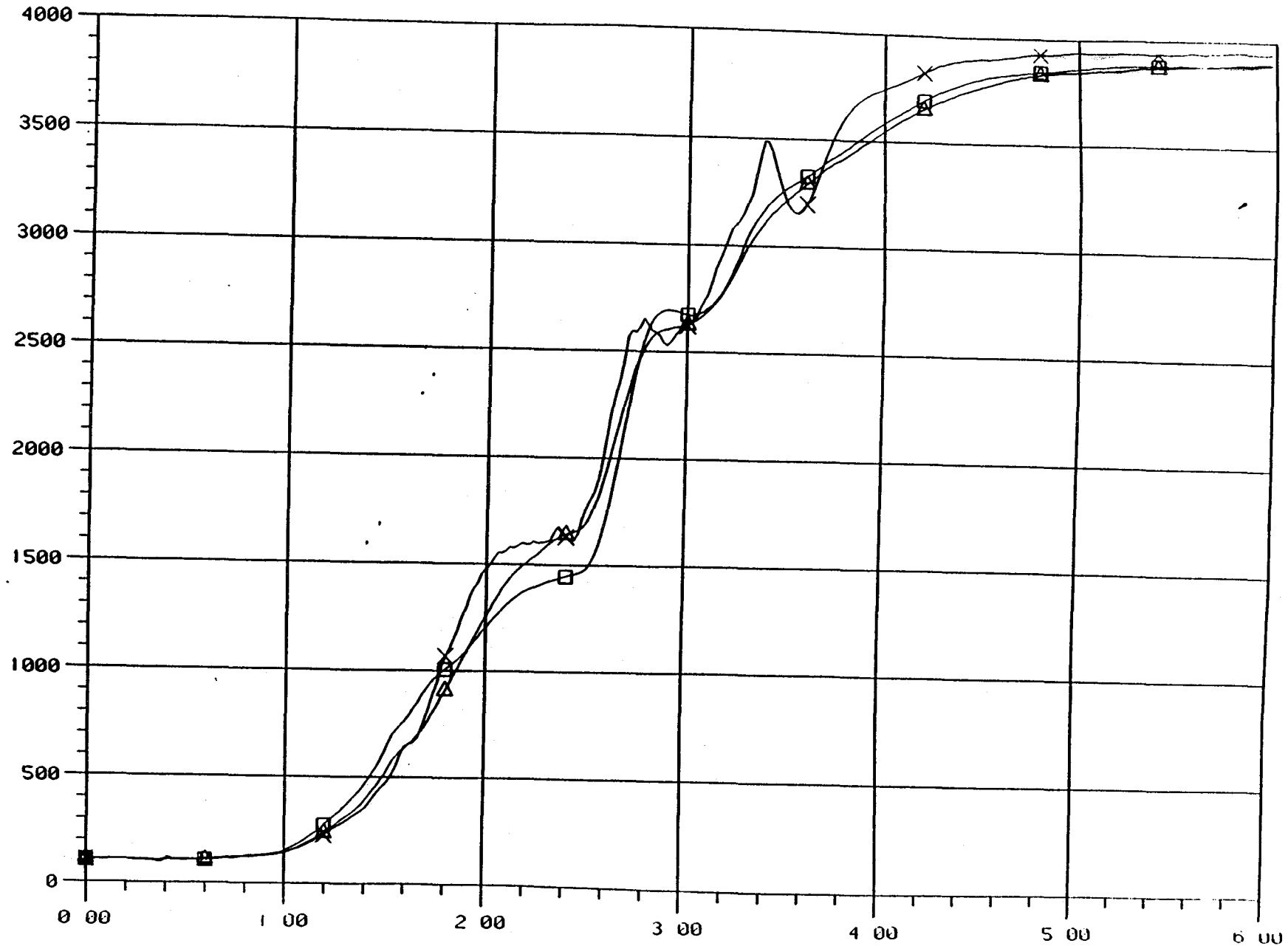
HPOP DS PR NFD 7K P

□□ TEST 9040117

334

HPOP DS PR NFD 7K P

START
P
E
R
F
O
R
M
A
N
C
E



ENGINE 2206

TIME

VER 4 000

DATE 04 01 1971

11

A3-117

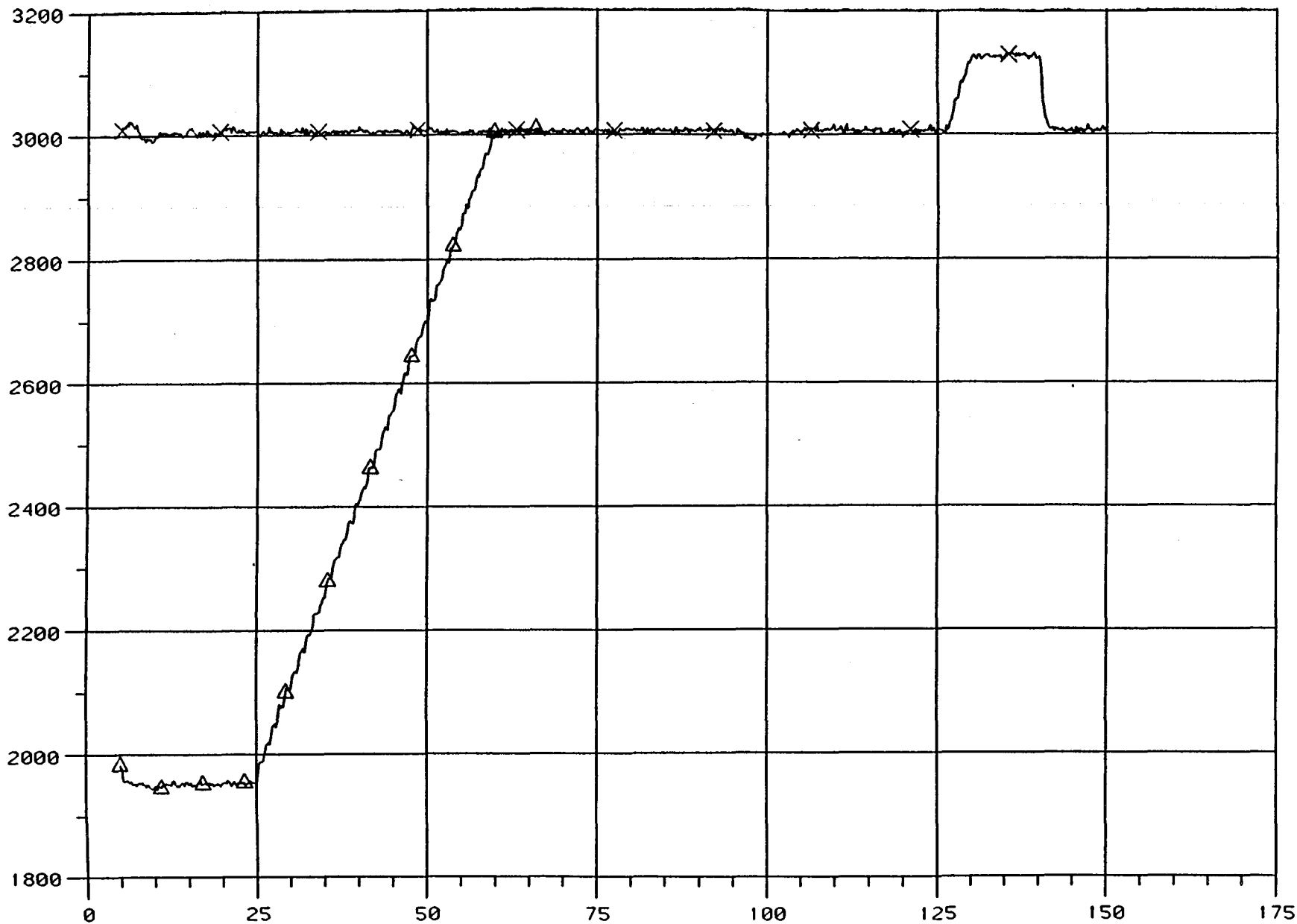
XX TEST 9040143
△△ TEST 9040142

63
63

MCC PC AVG
MCC PC AVG

CM
CM

MAIN STAGE
PERFORMANCE

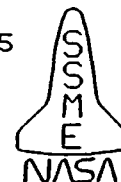


ENGINE 2206
SHUTDOWN

150 02 SEC

TIME FROM START COMMAND - SECS

VER 4 000
DATE 04/28/92
TIME 16 40 15

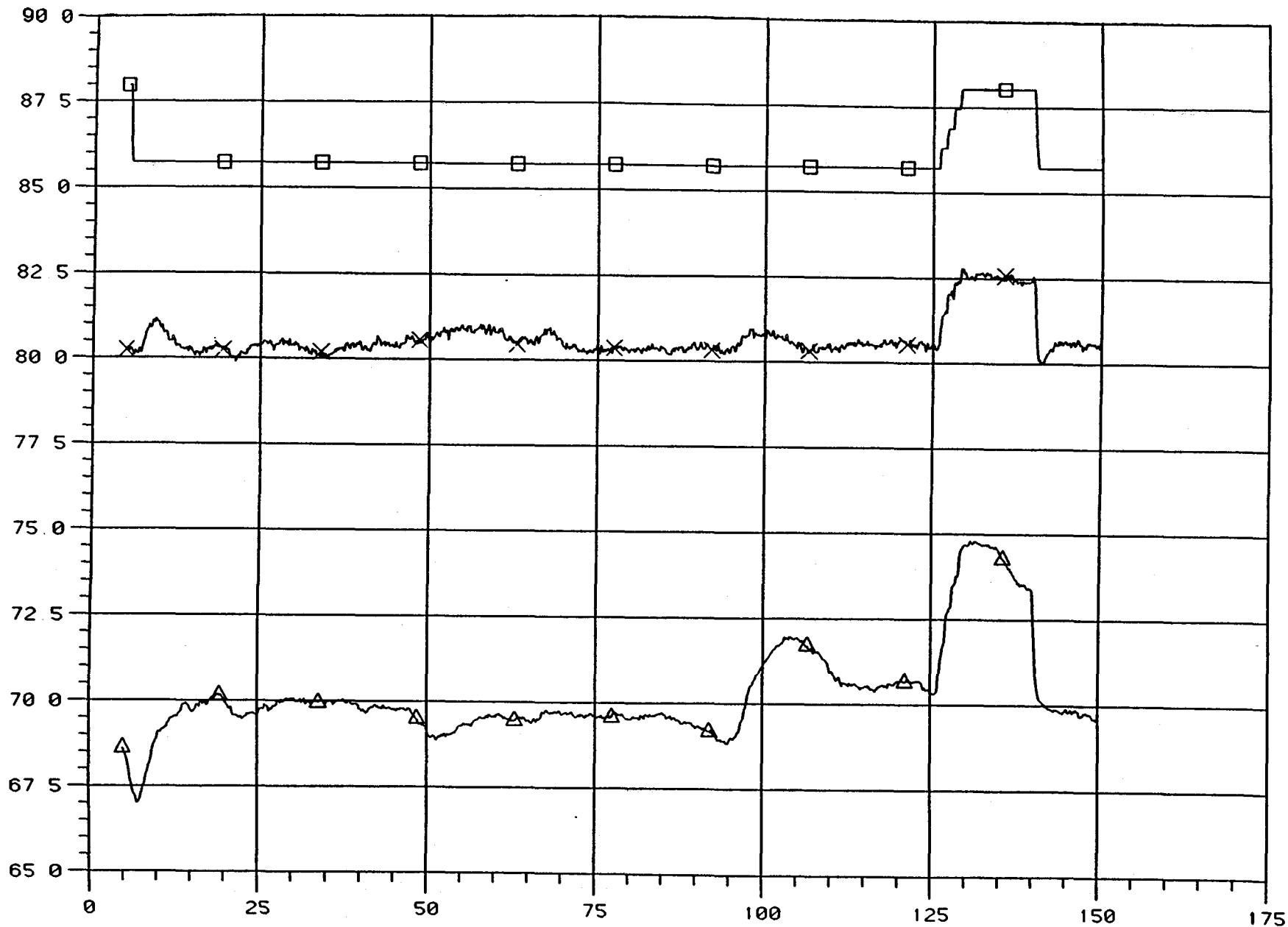


00 171

OPOV CMD LIMIT (R383)

PCNT

MAIN STAGE
PERFORMANCE

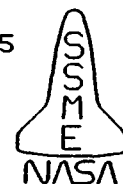


TEST 9040143
ENGINE 2206
SHUTDOWN

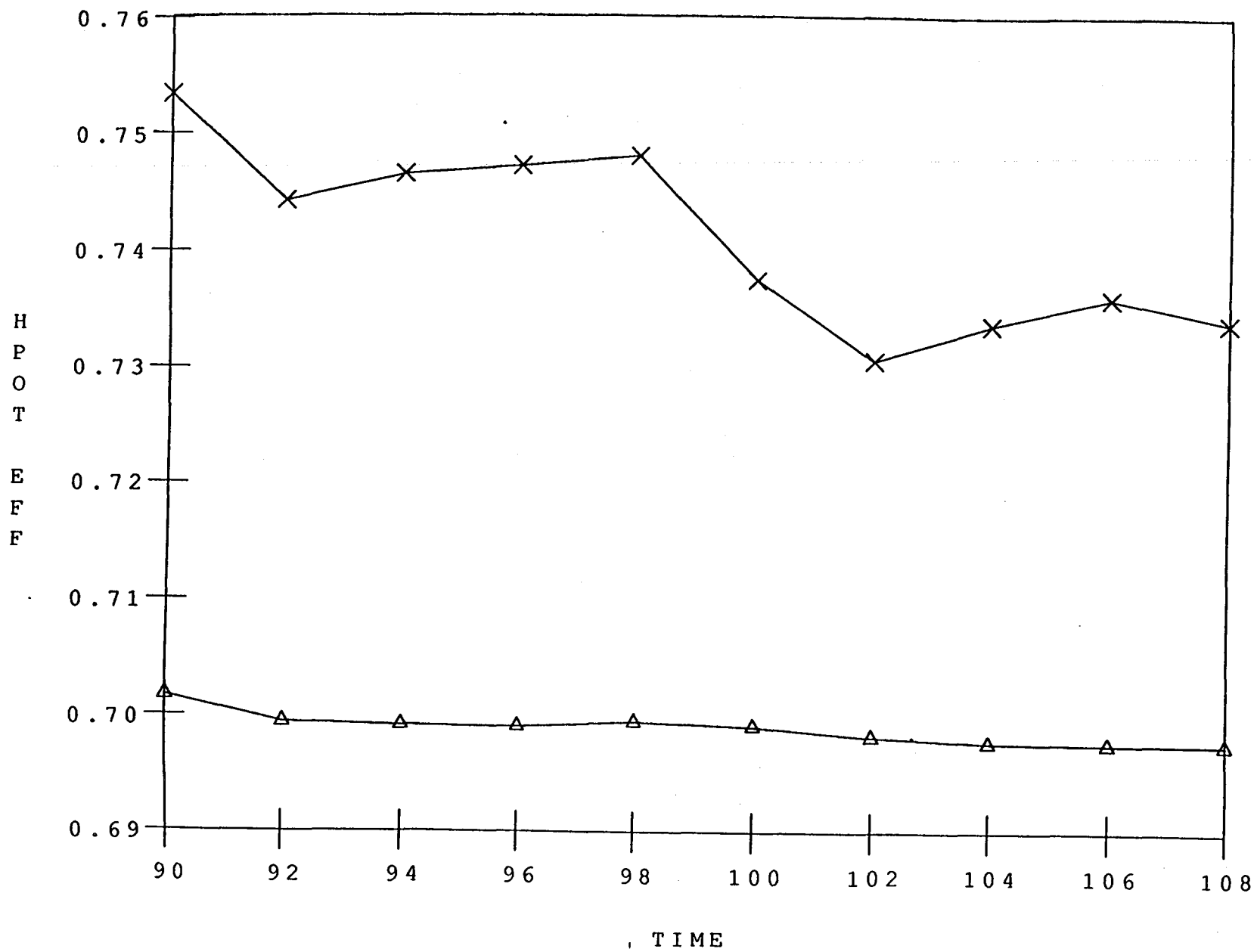
150 02 SEC

TIME FROM START COMMAND - SECS

VER 4 000
DATE 04/28/92
TIME 16 32 26



B1-143 TURBINE PERFORMANCE (ACTUAL & THEORETICAL)



Taylor Hooper Interview - A2-471 1May92

This failure involved the bellows internal structure. It connects the LPFP to the HPFP. It was the number 3 bellows, which is the third one here (see figure 8e), which is upstream of the flowmeter. This is what the internals look like. It's compressed and connected inside (look at figure 8f). The legs fractured and the loose piece went flying down the duct (see figure 8g). When the bellows expanded, you got leakage where the loose piece had ruptured the duct. The loose piece lodged in the flow straightener which is upstream of the flowmeter to help create laminar flow before going into the flowmeter. So the piece got lodged there, which was good. If it had gotten past the flowmeter into the HPFP, it could have wasted the engine. This rupture started a fire, which was what initiated the redline. The ambient powerhead temps exceeded the redline. That is a synopsis of the anomaly.

Catherine: Was it during Mainstage?

Taylor: Yes, it was during mainstage. There is a piece of film on it, and you can see when the bellows expand. The whole thing starts shaking. The acceleration data from the anomaly was phenomenal. Rick goes through that here in his report.

Catherine: What type of report is this?

Taylor: It is a failure report put together for the anomaly. Rick Ballard, Dave Vaughan and Larry Leopard were probably the ones that worked it. I wasn't in the group at the time. I wasn't doing data analysis at the time. He breaks it into two events. The first was where they first saw data skew. Where the first phase of the failure occurred. The second was at cutoff. He includes a time line in here. They were running along at 140 sec, had just throttled up to 104%, at 146.22 they saw an expansion of the bellows, that was when the leg broke internally. At 1/100th of a second later, the accel data indicated a spike. Then the HPFP inlet pressure started dropping off by about 10 psi.

Catherine: Because of the leak upstream?

Taylor: I believe it was because of the piece that had lodged in the flow straightener. At the same time of the inlet pressure drop, the inlet temp increased by .14, which he says was caused by the heat transfer due to loss of insulation at the rupture point.

Catherine: All of the ducts are insulated?

Taylor: Yes. Valve positions started to move. There was an increase in the FPOV position. You are losing fuel out because of a leak, the mixture ratio is dropping off, so you think that you should power up the fuel side to get your fuel back. But it actually sensed an increase in fuel flow caused by the piece that was lodged in the flow straightener. When the piece lodged in the flow straightener, it affected the dynamic flowrate, flow

A3-121

character, such that it thought that it was getting more flow than it actually was (See the report for complete details on this). O.K., the second driver was the fuel being lost through the hole in the duct. The cumulative effect was to reduce the FPOV command, Pc then dropped, the lox side powered up to maintain Pc. Turbine temps went up as well, by 15-18 degrees. Then the ambient powerhead temp sensors detected the fires which initiated the shutdown. He has some good plots in his report. They looked at every data point on these plots, because of how quickly it occurred. Rocketdyne put together some info in the pre-test report for the next test. They did a CFD model of the failure and included it in the package. They also included info about the hardware inspections which followed the test, assessing the size of the rupture.

Catherine: Do they normally inspect these ducts?

Taylor: No, we don't normally inspect ducts, because we very seldom have problems with them. If we change something, we replace the entire duct, like when we replace a flowmeter, we replace the entire duct, because they are all one piece. They can remove the flowmeter from the duct, but they normally don't. It comes assembled, and they usually leave it that way.

Catherine: O.K., let me read through this report, and I will probably be back to talk about it some more with you.

MARC NEELY
Component/Gains Analysis
4-29-92

First we can look at each component in general terms and then we will pick the main parameters that would represent that component. You are familiar with what controls the engine during formal operation so we're are not going to look at hydraulic lock up.

During normal operation the main chamber pressure and fuel flow are your two primary drivers. This is where I have trouble going to components because these two things influence everything going on in the engine. If chamber pressure goes up and goes outside its band versus its reference command then the OPOV closes down. If the fuel flow meter says based on Pc I'm getting to much fuel to maintain the right mixture ration then FPOV closes.

The FPOV has a cross feed gain. It follows the OPOV at a certain ratio. If the OPOV opens up 10% the FPOV opens up some percent portion of that. That's because it realizes its going to open so it goes ahead even though the mixture ration has not changed. That 's the heart of it.

Now I guess the next way to look at it is on a test to test comparison. Let's assume we have a baseline engine and let's change some component. That component is different in this respect and how does the engine balance in order to compensate. We have a baseline engine and for the next test we put in a lox pump that has a lower turbine efficiency. What that means is we must open up the OPOV in order to get the same speed out of the turbine to be able to give the same head you had on the previous test. When you open up the OPOV that gives more lox flow into the lox preburner and that will be registering higher chamber pressure to get the same speed out of the lower efficient turbine. At the same time the FPOV will have to open up a little bit for a couple of reasons. One is this being open more it is not getting the same amount as previously and the FPB discharge pressure reads a lower pressure because of lower back pressure (lower resistance in the system). I think that's it in a nut shell. I don't think you will see any other significant deltas. If the pump end is less efficient I think you will see essential as before but a higher speed to get the head rise but again that depends on the relative magnitude of change. You may or may not see anything else in the system. If the fuel turbo pump is less efficient then the opposite happens the OPOV opens up to get more lox and you see increase in that pump speed and may see increase in OPOV.

If the turbine efficiency is down you will see an increase in the FPOV position to get a higher fuel preburner chamber pressure to maintain a equivalent speed to get the same head rise across the high pressure fuel pump. If the pump efficiency is down its just like the lox side.

Now let's say we go back to a baseline engine and we change out a duct. Say we change out the high pressure fuel duct that runs between the discharge of the high pressure fuel pump and the MFV. We have two configuration ducts and as far as I know we are running both. They have a different inner diameter and you see a lot of difference in the reading of the HPFP discharge pressure. It's a static pressure and I believe that the smaller diameter one is the newer one and it has a lower static discharge pressure. Assuming we have the same duct if your put a duct on that has a higher fuel side resistance that will cause you to have to put more energy in the fuel preburner to turn the pump faster to get a little higher head rise across the pump to over come the increase resistance. That's if you have a smaller diameter your discharge pressure goes up.

This is something that has happened over on the lox side in the last year. We put on a lox duct with a 7 sigma high discharge pressure associated with it. It was because it had MR condition which is a weld condition. It was just a high resistance duct and it caused the lox system to work harder to overcome the resistance to maintain the lox flow.

Going back to the schematic.

The fuel comes in to the low pressure fuel pump through the 3 stage high pressure fuel pump and after it leaves the main fuel valve it splits and some of it goes to the main chamber, runs through the chamber, takes the heat out of it and is discharged and runs back through and drives the low pressure fuel turbine which turns the pump. In flight configuration this pressurizes the fuel tank and on the stand it is just burned and goes to a burn stack. But that is only a small proportion and the rest goes and splits. It goes into both ends of the powerhead. The powerhead has a liner, an outer shell and a liner. This turbine exhaust of the LPFT dumps into the liner and cools the whole powerhead and then dumps. There are two face plates, primary and secondary face plates and it all dumps out into the liner in between these two plates. It mixes and goes down into the chamber through the porous face plate. It also has holes

drilled around the outer circumference where it dumps fuel down the outer walls of chamber.

The second flow split goes through the chamber coolant valve. The third one goes down to the nozzle. It goes all the way down to the end where there is an aft manifold which is a big pipe that runs down the exit. It dumps and then it runs up 600-900 separate tubes that comprise the nozzle. It then goes up and mixes with fuel that goes through the CCV and they burn in the preburner. It is then exhausted through the turbines and goes through the injector elements and then burned. The chamber coolant valve is kind of in my way of thinking backwards. If you close this valve you get more coolant to the main chamber. If you open it you get less coolant to the main chamber.

On the lox side the lox comes in through the LPOP (let's set the pogo accumulator aside for a moment, it is just a suppressant system to take the pressure oscillation out of the lox flow). If you start pulsing in the lox flow it can show up as a pulse in the chamber pressure and it can couple with your control system. Your chamber pressure can get a pulse of lox and your chamber pressure can go up and your OPOV can slow down. Then it comes down because of the pulse and it says speed up and they can couple and you can get out of control.

The flow goes to the main pump. It is discharged and some of it splits off and goes back up and into the turbine. The drive for the LPOT turbine dumps back into the discharge of the pump so the turbine inlet and the pump inlet both go into the pump discharge. Somewhat like a closed loop. Some of it goes through the HEX. There is a check valve (an anti-flood valve). It goes through the HEX and is used to pressurize the lox tank. It is also used to maintain ullage and to pressurize the pogo accumulator and then some of it splits to the preburner pump. It enters axially and that is boosted up to feed the preburner through the OPOV and FPOV. The rest goes through the main lox valve is injected and mixes with the fuel.

A lot of times we will see cracks open up in the main combustion chamber. We've had some that were close to the face plate where we have lowered the resistance in the leg and have gotten an increase flow which causes an increase in LPFTP speed.

A3-125

Components, observable parameters for that component, and what drives them:

LPFP:

- . Turbine discharge pressure - fuel pressurization interface pressure

- . Turbine inlet pressure

- . Discharge temperature from the MCC - close to turbine inlet temperature

- . Speed - delta p across the turbine

- . Discharge pressure - changes in speed, changes in inlet pressure

- . The delta p changes when you go from max/min on the repress. Opening a valve controls the repress flow. This changes the turbine delta p which changes the speed.

- . A specific phenomena associated with the piston ring seal is in the turbine discharge leg where a seal unseats and gets more flow. Reduces the back pressure on the turbine changing the turbine delta p.

- . Things that can go wrong with pump: insufficient head rise, not enough discharge pressure generated. Low discharge pressure. (Neither are typically seen)

- . Things to consider if pump changed: No significant changes in the engine. The HPFP is very insensitive to the discharge of the LPFP until it reaches a point that it causes it to start cavitating. Must be a pretty bad pump to do this. The effect of the back pressure of the turbine on the chamber coolant. Chamber coolant temperature should be maintained well below 500 degrees. Turbine can act as a controlling orifice on the chamber coolant flow. It is not suppose to because there is an orifice at the discharge of each chamber that is sized to control how much flow it is getting.

- . Performance shifts seen in the LPFT in the past have only been attributed to piston ring seals in the main chamber powerhead. It allows more or less turbine discharge flow which changes the delta p on the turbine.

- . Change in the F-7 orifice may affect performance of this pump.

- . LPFP pretty consistent. Problems that have been seen are attributed to piston ring seal shift. This pump has very little effect on the performance of the engine.

LPOP:

- . Pump discharge pressure

- . Pump inlet pressure

- . HPOP discharge pressure

- . High pressure pump more sensitive to low pressure pump than on fuel side.

- . Tends to cavitate during shutdown because you back flow.

. Every time you drop inlet pressure even a little it must maintain the same discharge pressure. Has a constant resistance, constant flow so pump must speed up for the loss of inlet pressure.

. During vent the lox pump must make up the extra pressure, OPOV will open, the lox turbine temps will increase significantly, and the fuel system will decrease.

. Efficiency changes are seen when you change out pumps.

HPOTP:

- . Turbine efficiencies
- . Pump efficiencies
- . Turbine inlet pressure
- . Turbine inlet temperatures
- . Turbine discharge pressure

Test with ATD development pump.

Looking at a anomaly where it looked like there was a turbine efficiency change where the discharge pressure started to fall off which caused the discharge pressure to fall off on the preburner pump which caused the MCC pressure to fall off. As soon as all of this happened the OPOV opened up and caused the pump to speed up. What happened the speed fell off so OPOV opened and brought the pump back up to the speed at which it had before the anomaly but the chamber pressure was higher.

What is the time interval from the time the efficiency change sifted and the compensation occurred?

This was a dramatic fall out. It was about 5 seconds before it recovered. You would not be able to tell which happened first, the drop in chamber pressure or the increase in OPOV. Except you know the OPOV opened after the chamber pressure dropped. We see any number of duration of efficiency shifts.

How do you define efficiency?

I was referring to the fact that it required more power to get the same speed. We were getting same head rise for the same speed out of the pump end. That is how I arrived at turbine efficiency increase. It could be that there is something rubbing at the pump end. It still could be a hardware anomaly in the pump end. When you have an increase in the main pump speed you have an increase in the injection temperature because you put more energy in the lox. There is only one temp sensor in the main lox flow which is in the lox dome. You also have a preburner pump discharge temperature.

This is an ATD pump and not a RKD SSME flight configuration pump. They are usually more instantaneous. If it is a true efficiency shift it doesn't return to the same point as before the shift. We have some where there

is a shift momentary change and everything returns to be completely like they were before the event. Then sometimes they return to a slightly different operating point. That is what happened here. We are requiring a little more energy in the turbine end to get the same speed. But with the same speed we are producing the same flow. PBP discharge pressure was lower after the event than before. We had the same lox pump discharge pressure, a little lower preburner pressure with about 1.5% more open OPOV. We didn't have the same back pressure on the preburner pump.

What do you expect to see if the turbine efficiency shifted down?

Turbine efficiency down I would expect to see the OPOV more open, the same speed if it is just a turbine anomaly, a little higher turbine temperature. Temps are not real responsive in all situations. Depending on the magnitude of the shift you may see some effects on the fuel pump. You may see it back off a little similar to a vent. If the pump efficiency decreased I would expect to see the same thing except I would expect to see a pump increase to maintain the same headrise. This goes for both the main and preburner pump.

What about the pump inlet pressure?

If it is constant but for some reason had a change in the efficiency and it started going down the OPOV would open. It would only be able to do so if we increase the speed. So increasing the speed would cause the OPOV to open and increasing the discharge pressure would cause it to close. Where it would balance I don't know. It depends on the anomaly.

Pump Bi-stability.

At lower power levels there is a situation on the PBP, a stall point at a certain speed, the flow goes bi-stable and you can have two different flows for a given speed. When the flow decreases the pump slows down the Pc goes down the OPOV opens the pump speed increases and pushes back beyond that threshold and it has too much flow. It begins to oscillate between too much and too little flow. New pumps are screened against this. They are tested at 65%, 64% and 63% PL at 10 second intervals.

What happens if the inlet temperatures change?

What you are trying to maintain is mass flow rate. Increase in temp causes lower density, you have to increase pump speed to get the same mass for a higher volumetric flow. Those are not real significant.

Problems seen in the lox pump.

We've seen increase resistances go through the lox leg and suspected ice in the injector. It causes the pump to work harder to get the flow out. This shows up in the discharge pressure, the speed, preburner chamber pressure, pretty much in everything.

Basically if you increase the resistance in the lox system or decrease its efficiency, the ability to pump against a given resistance, then the pump must work harder. The turbine temps are high the speed is higher and the fuel system powers down a little.

Problems have been seen with the intermediate seal purge pressure.

What other kinds of things can happen to the pump that can affect the rest of the engine?

Rotor moving can cause an efficiency shift. Seal moving with respect to the rotor can cause an efficiency shift. Hot turbine gas is discharged through 2 stage turbine disk and there are some by-pass flows. If by-pass flow changes you can get more or less efficiency of the turbine. We have balance cavity pressures that tell us the axial position of the rotor. You would have to ask the component people about this. And I think that is more of an art than a science. On either side of the pump there are cavities that take off high pressure.

After the flow comes in it splits in the center of the impeller and this is discharged. Some of that which is discarded is taken into a cavity and put back into the inlet. It is kind of a circular thing and done on both sides of the impeller. It is orificed and the orifice size changes with the moving of the pump so it balances the axial position of the pump. You can sometimes see axial movement correlate with efficiency change.

Are turbine and pump efficiencies basically the major thing that will change?

Yes, you may have balance cavity pressures that say yes physically the rotor shifted or had some movement at the time we had the efficiency change. This is especially true if you have a big effect on the discharge pressure.

HPFT:

- . Measured propellant flow on lox and fuel
 - . Speed
 - . Preburner Pc
 - . Turbine discharge temps
 - . Use preburner discharge pressures and temps to know what the lox supply inlet pressure is.
 - . A problem with this you don't know the flow split. .
- Look at speed vs head rise
- . Look at speed vs amount energy put in preburner.

A3-129

.If fuel pump efficiency changes you must put more energy in it then the lox pump will throttle back just a little.

What kind of failures have you seen?

On the backside of impellers there are parasitic flows which is essentially a cavity type situation where you take a high pressure discharge change and you flow it to the back of the impeller and then put it back in so there is circular movement. We seen changes in that flow when the sealing material (which is very soft) breaks up and the flow path can change. When that happens you usually see an indication in the discharge temp that your discharging more parasitic flow that has more heat in it. Has a very small effect.

On turbine efficiency shifts we have to increase the preburner Pc to get the same speed out of the turbine. Those may occur because of sheet metal problems. You have a lot of sheet metal in the discharge and have had situations where the sheet metal had been torn loose and restricted the discharge flow.

On one test the MFV was mis-clocked on the actuator. The actuator was saying it was 100% open when it was at 94%. HPFP had a decrease of about 400 psi in discharge pressure and the pump was spinning fast, increasing the discharge pressure and the lox pump was throttled down a little.

We didn't get past the start. It was misclocked opened and we got too much lox flow in the main chamber early. The chamber primed but the back pressure and chamber caused the fuel pump to stall.

We have also had cracked impellers.

Gains for changing out components.

. If you change out your pump the preburner chamber pressure that is required to drive the pump to get a flow is part of the back pressure of the fuel and lox system.

. There are variations seen with hardware change outs.

. The nominal variation is 20 to 40 psi for a pump change out. This is due to efficiency and back pressure.

. Very little difference in combustion efficiency from one chamber to another.

. Almost all chambers as they get older have lox post plugged because they are cracked. When you get up to around 10 plugged post you begin to see a decrease in combustion chamber efficiency.

. Your main pump efficiency doesn't affect you engine performance.

Leaks.

If you have a leak down stream of you flow meter then all the lost fuel flow must be made up in lox. Because as far as the engine knows the flow meter has already recorded that fuel. For 1 lb of fuel you lose

you have to make up for it with 3 lbs of lox to maintain chamber pressure. When you have cracks in the chamber and you dump in fuel you have a lose of fuel and the lox system has to power to maintain chamber pressure. The fuel system remains constant because the engine thinks you are losing lox.

Any leak downstream of the flowmeter or on the lox side will be assumed by the engine as a lack of lox flow.

When you dump fuel overboard is analogous to a lox leak. We test at max/min conditions where we have a valve that open and closes to different flow rates. (min=.2 max =1.2). For every pound of fuel you dump overboard you increase the lox by 3 lbs. You see a change in the lox system when you change the repress on the fuel side. You don't see much change in the fuel side with the exception of the LPFP. When you go to repress it lowers the turbine back pressure and causes increase in the delta p on the turbine and the pump speeds up. Repress on the lox side doesn't show up very much.

Are there any other efficiency type measurements made in the powerhead?

Just combustion efficiency.

Do you measure things like resistances?

We look for resistance changes. We look at them in terms of a discharge pressure compared to an average data base. This can fool you sometimes. If you have an engine that has had some changes it is important to look at delta p's. We have had situations where some type of foreign particle has gotten into the system, a tool, a rag or anything else. That's something to look at but we typically do not clog the data book with delta p's.

Are there any other major components that you change out? For example are all nozzles the same, are all throat areas and exit areas the same?

No, they are measured and the measurements are available. That doesn't effect anything but ISP.

Hypothetically what would happen if you had the same problem on the lox side as you had with the fuel valve(partially closed)?

MARC: If the valve could stand it the lox side would burn hotter. It would not know where the resistance came from. You would look at the delta p across the injectors, look at discharge pressure minus the dome pressure. We have had test where we had increase resistance across the injectors and we thought we had ice in the injector.

Anything ever happen on the pogo?

We don't have any data on the pogo accumulator itself. We have a pogo precharge pressure transducer that looks at the pressure of the gox that we are injecting into the pogo accumulator.

Dave Foust

29 April 1992

Delta Book and Hardware Changes

Dave: Sometimes we'll change out the LPFP and all that effects engine balance. Flying a high efficiency Lox Pump drives turbine temps down which causes fuel turbine temps up. Efficiency of pump and engine on which it green runs affects flight predictions. Last flight, the HPOP was green run on an engine with fleet leader leakage on the MCC, i.e. it dumps lox out through cracks in the MCC, therefore the Lox flow is increased to maintain the MCC Pc, causes all parameters to go up. That pump is then put on a flight engine (nominal), where the increased lox flows increase HPOP DS P by ~75 psi. After further study, it was determined that all parameters were running 2 to 3 sigma high, so predictions had to take all of that out for the flight engine.

John uses the Gains book, say if he sees a decrease in resistance in HPOP DS P, there is a resulting drop in the HPOP DS P, which drops turbine temps, affects valve positions, etc. This book was originally published in 1985 for phase I pumps.

Tim: We have a book at Aerojet called the Delta book. I think that it is the same thing.

Dave: I can get a hold of John's if necessary. As a matter of fact, I will be getting an updated version for this next flight.

Tim: Brian was telling me that he also tabulates alot of similar information from special PBM runs.

Dave: Yes, thats another tool that I use. We predict at 200 sec in flight.

Tim: Would you please back up and explain what you mean when you say that you predict something for flight?

Dave: Predict how the engine will perform at 200 sec into flight. JSC has the flight rules - they monitor engine and vehicle performance much closer than we do. They will call an abort when we would not have considered things to be running off nominal. They need to insure that we have enough lox to complete a mission. If we're running off mixture ratio, that could happen. Rocketdyne gives them the official numbers. Ours are mainly a confirmation.

Tim: We're mainly interested in post-flight/test. You get the data back and from my observations, alot of reasoning is based on change in direction of parameters. But at some point, this type of reasoning is no longer sufficient.

Dave: That is correct. We're looking at the data of the 3 engines plotted simultaneously. On FRF, we noticed from ground-test to FRF, all fuel pump discharge pressures dropped 100-150 psi. That is not necessarily a big deal but we investigate it any way. We

A3-133

discovered that a ground test instrumentation probe being removed from that duct caused the resistance to drop to effect that kind of a change. In flight we compare across engines. If we need to track something further, we'll go back and look at the last time that orbitor flew, or maybe even look at the green run for a particular component.

We plot alot more data, from tanking to engine cutoff. One concern on flight is cavitation on the HPOP driven by the LPOP DS P. Shutdown at 0g, where inlet P's are what the ET is providing. Cavitation is indicated by HPOP spd being high, while the HPOP DS P is dropping, ie no head. Also look for pops. We had a good post-shutdown pop last flight. They had to do a hardware inspection of the faceplate as a result.

Jeff: Prestart is much more regimented in flight than in ground test. Most is time based. PSN 4 starts at t-4min. You can recylce if valves aren't cycled. Events must occur, i.e. LCC's must be met before you can launch.

Dave: You have a 3 min He purge/hour on the fuel side during which time you are looking for leaks. You should see temps increase because you are blowing warm He. If temps don't increase, you suspect leak. Ground launch computer dictates everything. You start your APU's on the orbitor. You only have a certain amount of hydrazine to run these, therefore you're limited about how long you can go into a hold also. If this is violated, you can run out of hydraulic pressure.

Another phenomenon is drain back. Once we have finished filling the tank, the lox and fuel that remains in the lines tends to be a little bit warmer. The bleed valves are still open, so you're filling a small amount of fuel and lox into the engine. The temps will begin to warm up because of this. The temps will warm up into the start box and you have about 1 minute to start once you enter the start box.

Tim: Do you ever use the delta book to do a diagnosis of say, you have two problems and you are trying to discriminate between them but you don't know the amount of changes that they might have caused?

Dave: I don't think that we have ever used the delta book in our analysis. Now, the kind of analysis that we do and the kind of analysis that John does are kind of separate. John looks strictly at steady state and the flight rule. We are mainly concerned with the mixture ratio flight rule that is concerned with lox turbine temps, HPOP DS P, FPOV and OPOV. We look at those 4 parameters. If the engine operates far enough out of these bands, then we say that there is a mixture ratio shift that is caused by a flowmeter shift or whatever. There are several things that can cause it. They say based on this shift we may run out of lox or not based on whether the engine performance shifted high or low. It just depends on the residuals that we had for that launch. And thats

where JSC can call an abort or scrub.

So John looks at steady state and we look at the transients, the engine system health and performance more than steady state. We'll plot drifts of instrumentation as well as engine health. If we have a situation of something totally off nominal, such as turbine discharge temps, more than we predicted then we will be involved. There have been occurrences where pump efficiencies shifted, turbine temps have risen a few degrees

Dave Foust 29 April 92 Interview cont'd

We have actually had flights where we experienced efficiency shifts on one of the pumps. All of the sudden the turbine temps will kick up a couple of degrees, the discharge pressure will change, all for no apparent reason. We finally conclude that it was some kind of efficiency shift. If that happens before the 200 seconds, then our predictions will be off.

Tim: Perhaps we should back way up. What exactly is your responsibility during flight?

Dave: Helping John to make predictions for flight and then also to look at the actual flight data and compare it to our predictions and Rocketdyne's predictions.

Tim: And if there is any kind of anomaly, are you then in charge of investigating it?

Dave: Probably Darrell and I are the primary ones to look at that data. Everyone looks at it. Darrell and I are the ones in charge of putting the package together for the data review. John and Darrell interact very little. I do work for both of them, like putting together the 2 sigma charts as well as the standard plots for the data review. I do the packages for both people. I also keep alot of the flight data bases: the 2-sigma and the flight database.

Jean: Do you have a hardware change database?

Dave: Yes. I have a hardware change database for flight engines only.

Jeff: Are you going to port those guys to Ingres?

Dave: I plan to.

Tim: So what do you have in the Hardware change database?

Dave: It's is similar to the hotfire database, with extras. I track the 4 pumps on each engine, how long each engine has flown, test date and test number, hex orifice size, C2's and Kf's, as well as all hardware changes that have occurred after each test or flight. If they change out a flowmeter or a pump, I track that because it all has an affect on the performance of the engine.

Tim: Would you track the relative pump efficiencies?

Dave: No, I don't keep track of pump efficiencies. Brian was doing that at one time. After each test he would input the efficiencies that had been calculated for each of the pumps. But I don't know that he is still doing that. He would dump it all into a spreadsheet.

Tim: I often hear reference to "Well we have this one pump that was on this other engine and we know how it performed..." Is that information that is recorded, or is that something that is carried around in your head?

Dave: It's not written down. But if you see an engine or pump run totally different from the last pump, then you begin to suspect that the pump is bad. You can have poor turbine efficiency, you can have poor pump efficiency, you can have a great lox pump but a terrible preburner pump. Therefore your discharge pressures are different, your valve positions are opened up. There are several things that could be bad. You start to go through each one. You get Tracey, Brian or Bil to do a data reduction on that particular test. Then we'll look at how those numbers fall within the bands of pump efficiencies, suction specific speed, and all of the other parameters. So we don't go into a lot of detail on pump performance unless there is a problem.

Bill: I know that you don't use the Delta book for diagnosis. But the magnitudes, the ways certain parameters respond, would that drive your logic as to how you would diagnose? I mean, if there was a hard failure, you wouldn't investigate a sensor failure. If it was too small a delta, you wouldn't even know that it was an anomaly. Should the magnitudes drive the logic of your diagnosis? Is that something that would be useful for us to pursue?

Dave: Yeah, it does make a difference. That is a judgement call that varies from person to person. You may see a shift in pressure that may just be a run-to-run variation. Someone else may see it as a potential problem. For example, there is a lot of instrumentation that drifts on flight. A lot of parameters freeze up. Those are things that you have to learn from experience. Hot gas injector pressure is one that we have a lot of problems with. But you learn what to expect. Sometimes that measurement doesn't respond to the bucket or the 3g throttle down. FPB Pc is a typical drifter. It may drop as much as 7500(?) psi. We don't worry about it, but just note it as an observation. But that fuel pump discharge pressure coming up on FRF, that caught our eye. That is a big shift. You change out a fuel pump on an engine, you are going to see a change in the fuel pump discharge pressure, 40-50 psi. So you will see some changes based on hardware changes.

Tim: So maybe we can talk about some other changes. That was a fuel pump change, you said?

Dave: Yes, you would expect to see fuel pump discharge pressure change (40-50 psi), you would expect to see the fuel turbine temps change. You expect to see your FPOV change, because you require a different amount of power to achieve that discharge pressure.

Tim: O.K., how much would you expect to see your FPOV change before you would worry about it?

Dave: Oh, about 1-1.5% depending on the efficiency of the turbine.

It may take more FPOV position, more lox flow to get the same performance out of that pump as was on there before.

Jeff: What defines a bad turbine?

Dave: I guess you would have to look at the 2 sigma's. If you look at the FPOV 2-sigma and it's outside, then it must be a pretty bad turbine, or pump, it may be the pump because it requires more pump speed to achieve that same discharge pressure. Typically, we use our 2 sigma stuff to determine if something is a bad performer.

Tim: And in that case you would be looking at speed?

Dave: You would look at fuel pump speed. If it was really high, you would determine that it wasn't that good. Or you would look at valve position. If the preburner valve position was really high, then that is a really good indication that it is a bad pump. If the fuel pump discharge pressure was really low, that would be a really good indicator. Same thing on the lox side, if you have a really bad lox pump, you may have a really efficient main impeller on your lox pump. Since you have an efficient impeller, your lox pump discharge pressure will be down a little. But your preburner pump impeller is on the same shaft as your main impeller. So you've got a lower speed and not that great of a preburner pump impeller. So you're putting out a lower preburner pump discharge pressure. That's the pressure that feeds into your FPOV and your CPOV. Since your pressure is down there, you're going to crank your valves higher. It all feeds together. You can't change one thing without it affecting just about everything else in the system. So any time you change either of your 2 big pumps, you're going to affect the system quite a bit. Change your LPOP, you might affect the lox side some, but you won't ever see it on the fuel side. Change your LPFP, you might see it somewhat on the fuel side, just because it's in the coolant leg. Your MCC coolant flow from the discharge drives the low pressure fuel turbine. So if you have a different turbine in there, you can have a different resistance, your flows could change and that affects turbine temps and a few other things. But any time you change your big pumps, you see major changes. That is the hard thing about predicting. You have to predict how much the turbine temps are going to change. Another big problem is the delta between the A and B channel. We don't know if its engine driven or pump driven. We have theories on both, and neither one seems to hold true all of the time. And that can make a big difference, especially on this flight rule, because turbine temps are one of the things that they are trying to look at. And if you mispredict the delta, say you're predicting the A to be cooler than B by 20 degrees, but it ends up being cooler by 70 degrees, that makes a big difference. We can hit the average pretty well, but miss the delta completely, which makes your predictions completely wrong. That is another thing hard to predict, and it changes from engine to engine. Generally speaking, the fuel side is engine driven and the lox side is pump driven.

Tim: So these numbers that you have in you head, e.g., this 40-50

psi change, are things that you are looking for in the data that you will flag? If you know you have changed your pump and you see a change greater than that?

Dave: Then you start questioning. Is there something else going on here? That is why we questioned on that FRF thing, we had 100 psi difference on 2 engines and 150 psi difference on the 3rd engine. That was a lot to see a fuel pump discharge pressure change from the ground test of that engine. That is why we started investigating it. We just did not expect that big of a fuel pump discharge. We had changed out the big pumps on all 3 of the engines, but we did not expect that much of a pressure change. So that is when we started investigating it.

Tim: Last time I was here, I walked through an analysis with Randy, and there was a high turbine discharge temp on the fuel side. He came up with 2 hypotheses. One was that the pump was changed, we went from a Pratt to a Rocketdyne, and the other was that the F7 orifice was changed. Both of those could have effected that change. I then went and talked to Brian and asked him what effects changing the F7 orifice would have on the turbine discharge temps, and he said it would only affect it by about 50 degrees. They saw a much bigger change on this particular test so Randy ruled that hypothesis out as a possible cause. Is that something typical of what you would do?

Dave: Yea, that's pretty typical. With an F7 change, if you are running hot on your chamber, and you get back from flight and see that your chambers are blanching, cracking, you would have changed your coolant flow by changing your F7. That is a possible driver. However if you saw a turbine temp shift that you were not expecting, you would look at F7 orifice, LPFT or LPFP that would change the resistance in that line. What they do is block passages in the LPFT, to try and balance out the speed for the discharge pressure that they want. If you have more blocked passages on one than another, you have changed the resistance on that line, therefore the flow going through that line is going to change. That dumps into your fuel turbine and changes your fuel turbine temps, and that is probably where Randy was coming from. Going from a Pratt to Rocketdyne fuel pump, or either way, those two are a lot different. I am not that familiar with that type of hardware changes but I know that the resistances are a lot different, the bleed flows are different, there are a lot of changes that are going to rebalance the engine, with that one change. We interact with Brian's group a lot when we see something that we don't understand. For example, on this last flight prediction, when I saw the 75 psi difference on the discharge pressure from the green run of the lox pump (which is a lot different from the 15-20 psi delta that you might expect), we tried to determine if it was an engine driven parameter, because our predictions were really off. Because generally, even if you change the lox pump, the lox pump discharge pressure will stay the same. So when we green ran the pump, it was running 75 psi high on the lox pump discharge pressure, from what we usually fly. That is a big change for that

parameter. When you take out that 75 psi, it's going to change lox pump speed, turbine temps, preburner pump discharge pressure, valve positions, etc. Here we go to Brian and say how is this change going to affect our engine system. He will do a PBM run, which is really a gains run. That is what that Delta book is really, a bunch of gains, John could go in and say, ok for this 75 psi change, we see it affect the engine this way. The PBM does good on some things and not on others. It doesn't do good on speed, I don't believe the speeds that it calculates. Delta gains are pretty good. Valve positions aren't that great either. We're trying to use 3001 data to improve the data.

Jeff: What do you consider to be a good prediction, within 5% or 10%?

Dave: Probably on turbine temps, a good prediction would be within 10-15 degrees, and when you're talking temps that run around 800 degrees, that is pretty tight. On the speeds, I have a looser interpretation than John, you are talking 35000 rpm, I figure that if you get within 100-150 rpm you have done pretty good. John likes them a little tighter. The same on the lox pump speed, it's running about 28000 rpm. The LPFP speed is only running about 15000 rpm so you should be able to get within about 50 rpm there. LPOP is about 5000 rpm, so you have to be even tighter there. For valve positions, you need to be really close. The flight rule on valve positions is real tight, you need to be within .5%. So your predictions need to be pretty close. For the flight rule you can violate up to 4 criteria, and usually the valve positions are the ones that you are closest on, because it's really hard to predict them. They are transient, adjusting to Pc changes as well as other changes. Even if you hit it right at 200 sec and it happened to shift down, then you have to go back and look at all of the data for the entire flight. LPOP discharge pressure you ought to be able to hit within about 10-20 psi, LPFP will vary a little more, say 20-30 psi. They like them as close as possible. The sigmas don't float in my head. We have a database of all of this, but only for flight. We have been keeping it since Challenger, so now it's data is beginning to be pretty representative. We have changed alot of things for flight. For example, we now have fixed orifice on lox side for repress. We used to have a valve to shuttle max to min repress, now its fixed. It changes things a little bit as far as how you feed back in with you inlet pressures. We have a different hex orifice, we have changed to inconel ducts with smaller diameters which changes your fuel pump discharge pressure. These are all changes to account for. The new hardware changes things.

Jean: Do you include all hardware changes in your database?

Dave: I try to, flowmeters, pump changes, orifice changes, duct changes.

Tim: Now you are talking flight engines, so the change would be from between acceptance test to flight?

Dave: Or from the last flight of that engine. Once you acceptance test the engine it will fly 6,7 times. Then they may go rebuild it. For example, 2017 is one we are going to fly next, it last flew before Challenger, and they have since rebuilt it, but it still has the same engine number. Since it has completely new hardware, they had to acceptance test the engine again. A lot of engines will fly 4 or 5 times and then they will change the hardware.

Catherine: Do we have this same type of database for ground test?

Dave: Yes, in Tracer. But locally, we don't have any of it. Tracer tracks pumps, seals, builds, everything. There is too much diversity in ground test engines, you have 2 vs 3 duct engines, you have the work going on B1, the ATD work, A2 is the only one devoted to flight and green runs.

Tim: When you talked about changing a fuel pump out, you talked about seeing changes up to a certain amount. Are those limits reported anywhere, or do you infer them from your database?

Dave: You could probably infer some from your 2 sigma database. Based on hardware changes you could probably pull out some numbers. But a lot of it just depends on what you have seen in the past for particular parameters. For the types of things that you guys need, we could probably put together some numbers, but we haven't done it before. It's mostly in people's heads. When we had that fuel pump discharge pressure change, we did go back to the 2 sigma database. We tried to see what kind of average value you would get from that hardware changeout. You want to backup your feelings about what is causing the changes.

Tim: So when you initially look at the data and note that the discharge pressure was high, and you know that you changed the pump from pre-test, when you first pass through the data, is that enough of a mental note, to say that you will go take a look at the actual numbers to see if they fit that?

Dave: Yes, we all look at the data and we compile a list of all the things that we have seen. Then we start looking at the individual things, why each thing happened. The fuel pump discharge pressure was off, why? Then we go back and look at the green run and see how it ran there. We go see all of the hardware changes that were made, was there a shift in performance during the flight, did turbine temps drop for any unexpected reason? You start doing this type of reasoning after you have gone through all of the data. Then you start investigating, making special plots to examine.

Tim: But if you changed a pump and you only saw the discharge pressure go up by 20 psi, would you make a note of that?

Dave: Probably not, because it's not enough of a change. That is a gut feel type of call. Those types of calls vary based on how

A3-141

much you look at the data, how recently you have looked at the data. That is why we have several people look at the data, rather than just one or two people. If you don't look at flight data alot, you may flag things that would be investigated for ground test. Darrell and I would more than likely pass it over. There are certain things on flight that you don't see on ground. Flight data tends to be cleaner, because you have really good hardware. We tend to knit pick it, because we generally don't have anomalies, we can't. So we pick out some really small things to investigate. One interesting one, epoxy resin got into one of the MCC sense lines. We have 4 lines. There was a delta between the A to B channel, which caused to engine to run off nominal because it wasn't sensing pressure correctly. We noticed it, and analyzed it, knew that it was real, and flagged it. In the post-flight inspection they found residue and were able to explain it. You have to take into account the accuracy of the instrumentation. To measure the 3500 psi in the chamber you use a 5000 lb transducer. One percent accuracy on this causes a flag for 5-10 psi delta, is this significant? The same thing is true for the speeds, when you're talking 35000 rpm, is a one or two percent shift significant?

Tim: So briefly, I know you change every piece of hardware. What are the biggies?

Dave: Hex orifice is changed to affect hex interface temp. We are required to provide a certain pressure and temp to tank ullage, the fuel and lox have certain temps and pressures that they cannot exceed, they also have mins. F7 orifice is to change the amount of coolant flow going through MCC as well as LPFP speed, there is a speed limit on that. Actually it is a flowmeter speed that is sensed in that line. Those are the main ones: Inconel duct, 4 pumps, hex orifice and F7 orifice. They change tons of other things that we aren't interested in.

Jean: What about seals?

Dave: We aren't generally concerned, except for the Intermediate Seal, it has an LCC on it. It is one of only two seals that have an LCC on them. They try to keep lox and fuel from mixing, so they keep a He purge on them all of the time. It's N2 in PSN1,2 and they switch to He in PSN3. There is some minimum value that it must maintain, 170 psi. You can't launch if its lower than that on that seal. You want to insure there isn't any mixing going on around that seal. And that also depends on how old the seal is. A brand new seal will run at a higher pressure because it is tighter. If it's been run alot, it will have a lower pressure. You have to meet those on green run too. Typically, for a questionable pump, this would have been addressed on the green run of that pump. Rocketdyne would have answered a RID (Review Item Discrepancy) from NASA. They would have to justify why it had occurred before that pump would be accepted for flight. So flight personnel get involved in green run and acceptance tests for the engines and pumps.

Tim: Would seal changes be noted on the pre-test?

Dave: It would be noted in the Pump build letter. Typically when you do something like that, they roll the revision number on that pump and that's how you know something has happened. It's the R part of the number, 0810R2. Every time you change out a non-rotating part, you roll the revision number.

Jeff: So when you note a revision number change, you have to go to another piece of documentation to find out what exactly happened?

Dave: Yes.

Catherine: Do those build letters also record leakage rates for those pumps? Or any torque testing, etc?

Dave: Yes. They also do post-test inspections where they do this type of testing where they record leakage, running torque and breaking torque. They have to explain why if they get odd values for any of these. You have to pass a lot of criteria on the green run before you're accepted. Then Rocketdyne puts together a pump acceptance review package that they present to NASA at which time it is accepted or sent back for more testing or any RID's satisfactorily answered.

Tim: Any other hardware changes that are of interest?

Dave: Those are the major ones, but they may change the nozzle, that doesn't affect engine performance that much. It will affect Isp, that is specific to throat and nozzle areas. It won't affect turbine temps or whatever, but it will affect overall engine performance.

They can change out MCC's, that changes your combustion efficiency. They change out powerheads, but that is a very rare changeout. Nozzles aren't that uncommon between acceptance test and flight. Flowmeters are changed. If they are calibrated, that won't affect you, but if not, it will affect your engine. Flowmeters are sensitive to engines, to stands. We can calibrate it on A2 and run it on A1 and it will run off nominal. We then get a new C2 and Kf.

Tim: Why is that?

Dave: We think that it's because of the different stand configurations. But no one really knows. They change out flowmeters fairly regularly. But you will get C2 and Kf changes to go along with it.

Tim: Will instrumentation changes affect you?

Dave: They are pretty transparent. There is one pressure transducer, the Staped(?) vs CC, and it tends to run low. Otto is adamant that only CC's fly. The Staped's tend to drop your pressure by 7 to 15 psi. If you are already running on the low end

of your acceptable pressure, as on the Intermediate Seal Purge pressure, then the pressure transducer will put you below the acceptable limit. The pump won't be accepted. The redline parameters all fly with CC transducers. That is the only instrumentation that we are sensitive to.

Tim: Do they ever screw up on the calibration constants?

Dave: Oh yea, very often on ground-test, very seldom on flight. You get garbage data. Sometimes you can put in the correct constants and recover the data, other times its non-recoverable. We've had fuel turbine temp and lox turbine temp transducers go bad on us during flight. Some of the temp sensors will debond during flight, so we start getting erratic data and then it goes to ambient.

Bill: You said that when you change out a nozzle, you don't see that many effects. Would the resistance in the coolant circuit be affected?

Dave: Yes, you would see it a little. Where you would really see it is the MCC coolant discharge pressure and temp, because it is in that coolant leg. But you won't see a big difference. Mainly, you see Isp shifts, unless you have a real leaky nozzle, but you wouldn't put that on a flight engine. A leaky nozzle is the same thing as dumping fuel overboard. It would cause increased lox flow. Nozzles on flight are usually in pretty good shape.

Tim: Well, we wanted to get a feel for two major topics: one, the Delta book, gains type of reasoning, when you really become concerned about magnitudes of changes in the diagnostic process, and the second was hardware changes.

Dave: Yea, as far as that Delta book is concerned, we don't use it that much in diagnosis, we use it more for predictions. John uses it, Brian may use it. It's basically hundreds of power balance runs that were compiled years ago.

Tim: In analysis, you prefer to use your 2-sigma database?

Dave: Yes. There isn't alot of confidence associated with the PBM. There are alot of empirical equations that don't match up. It is literally a power balance. The power may balance, but the mass doesn't balance out. They don't use all of the fundamental equations. And the mass numbers won't make sense when you do that. Speeds don't predict very well. John is trying to do that with 3001 data.

Bill: But didn't you say that it does all right with the deltas?

Dave: Yes.

Bill: So you can rely on the relative values? Would you feel comfortable with the relative efficiency values?

Dave: I would feel more comfortable. For a flight, if we changed out the fuel and lox pump, I would use the pump multipliers. We have one for the fuel pump and the head coefficient, we have one for the LPFP, the LPOP, head coefficients and efficiencies of the main impeller and the preburner pump impeller. There are the 4 multipliers that we use. We get the green run info and the pump multiplier for that particular pump then we get a number for the pump that flew the last time. We then put the old pump and it's multiplier, the new pump and it's multiplier into the PBM and do two different runs. We will then get two different speeds, calculate the delta between them. I am real comfortable with that for making flight predictions. It's a little bit different if the engine has not flown before, or you have changed out a pump. We adjust those numbers as necessary.

Tim: I have a better feel for this.

Dave: I know that you are going to need specific numbers for your project. We will have to sit down and work those out for you. It's not something that we currently have written down.

{Tape 5 after Marc Neely conversation}

{Begin conversation with Dave Foust, June and Claudia}
{Comparison tests}

....is a thermally affected measurement that tends to drift down. They have a remote mount that's supposed to solve the problem. We can go through a databook or flight review or whatever. Typically on a flight review we'll pull out all the instr. that are drifting or erratic, whatever.

Q: What would be ___?

A: We can go through a flight review, ground, whatever.

Q: Ground would be better.

{June is showing the system architecture and explaining the modules}

A: As far as the feature extractor, it's probably easier to go through a databook and have someone point out what drifts, etc. We could talk about the comparator now. On a ground test, we typically compare to the same engine. A lot of times the choice of comparison tests depends on the stand. For example, A1 is right now dedicated to the phase2+ powerhead. So, for this stand we always compare that engine with itself or a prior phase2+ engine. Otherwise you wouldn't have a good comparison.

Q: In general, will all parameters be different?

A: Phase2+ dramatically affects the fuel turbine temp distribution (the spread between the A and B channel) because you're changing the way the flow is coming out of the turbine and going into the hot gas manifold. Also, the coolant circuit. Phase2+ has a history of burning up chambers because of this. We open up the f7 orifice (in the cooling line that goes through the MCC). The flow comes into the chamber wall, goes through this f7 orifice, and that flow goes down and drives the LPFT which powers the LPFP. Once through the LPFT, some is bled off to the repress system. The rest of the flow goes to cool the bearings, PB wall, etc. and eventually dumps into the turbine discharge and down into the hot gas manifold where it's burned up.

Q: They open up the orifice to reduce the resistance?

A: Correct. To increase the flow, you open up the orifice and cool down the MCC. The other thing they do is open up boundary layer coolant flows. You have a primary and secondary face plate on all injectors. On the main injector, your lox post sticks through the primary. The primary face plate is down here where the actual combustion takes place. The hot gas manifold flow comes in between the face plates. This is H2 rich gas, going into some holes inside the lox pumps. Then the lox comes down through the post and the H2 swirls on the outside edge, and they mix at the tip in the actual combustion chamber. What happens is they have holes all around the circumference of the primary face plate, they dump H2 in there to cool the inside walls of the chamber. When you crack the MCC, you're not getting enough coolant flow on the inside as well as through the jackets. They enlarge these boundary layer coolant holes to allow more H2 to go down for film surface cooling on the inside wall of the MCC.

Q: How do they enlarge them?

A: I think do it electrically. These are very small holes. Electrical discharge machining (EDM), Maybe?

Q: This is something they would do before the next time they'd run (the engine)?

A: Right, if they were cracking real bad in one area (typically don't crack uniformly) they would enlarge the coolant holes above that area to allow more boundary layer flow to come and cool that

A3-147

area.

Q: They know where they've cracked it because they've looked at it visually?

A: Yes, they do visual inspections after each test. They map out where the cracking is in relation to circumferential locations. On some bad ones, they'll even go in and try to close the cracks up manually. This is called pening them shut. Like a pening hammer.

Q: Can they see that in the data at all?

A: You see the results of it in the data. You really don't see "it" because it's essentially burned up and spit out the nozzle. The way you do see it is that it's downstream of the flowmeter, so you're not burning this fuel efficiently (not getting ISP and chamber press). It's the same thing as a nozzle leak, the MCC pc has to be maintained, so they increase the lox flow to compensate for the fuel dumped overboard.

Q: That's the fuel that's doing the film cooling?

A: That's part of it, some is the channel cracks. It's pretty high pressure, about 4000 psi.

Q: So you can distinguish between nozzle leakage and an MCC crack?

A: Basically, you do it by inspection. It's about a 3 to 1 ratio. If you're leaking a pound of fuel, either through the nozzle or MCC, that equates to approximately 3 pounds of LOX. If you have a 9 crack chamber and ten tests later the LOX flow has gone up about 6 pounds, you can estimate that about 2 pounds of fuel is leaking (whether it's through the nozzle or MCC).

Q: If there's more flow going through "this" leg, is it going to affect the shaft, etc?

A: Yes, the LPFP speed will increase, thereby increasing the HPFP inlet press. It goes all the way down and even affects the turbine temps because you take the bleed flow off to the repress, this other flow goes into the PB chamber walls into the bearings, cool "this stuff" and dumps back into this flow downstream of the turbine. So, if that flow happens to hit right on a turbine discharge temp sensor, it can actually cool it quite a bit. You have a lot of swirl and turbulent flow so it's not simple, but it can cool down the turbine temps and cause problems there.

As far as phase2+, other things that might be different...

{Recapping what he told people earlier in the day about h/ware changes... a philosophical discussion}

A2 stand is used for a development engine, or an acceptance test for a flight engine. On a development engine, we typically green run flight h/ware (The four pumps, actuators, valves, etc). For comparison tests on a green run, you want to pick the same engine. Engines have their own set of built in resistances and you're trying to use those. So, typically on a green run, you pick either the last time that pump ran or (if they rebuilt a pump) you want to compare this pump vs. the last pump. If it's the same engine, fine, otherwise you have to use a different engine.

Q: You want to compare a rebuilt pump to the last time the pump ran before it was rebuilt?

A: Correct.

Q: On that same test stand?

A: If possible, and on that same engine if possible. But sometimes you'll green run a pump on A1, sometimes on B1. Right

now B1 is dedicated to the PW pump. They'll also run the Pratt pumps on TTB.

Q: How long is it going to be before the phase2+...

A: I think the phase2+ has been accepted and certified for flight, but there's concern about it - is it going to burn up chambers. Chambers take a long time to make and there aren't a lot of chambers to burn up. They're trying to make fixes to alleviate that problem. Eventually I assume the Pratt-Whitney pump will make it to all the stands, any engine. That may be a good way away, however.

If I was going to make a comparison for a LOX or fuel pump that we were going to green run, I'd ask how that pump ran last time. I'd take a test with that pump (it could be rebuilt, it may be another one) I try to compare it back to the test it was run on before. If it's the first time it's ever run I'll compare it to the last time the engine ran. You try and get as much in common as you can. The second consideration is profile. Most of the green runs are fairly standard. The engine acceptance profile is 550 seconds, and is primarily to simulate flight as much as possible. So, we pick comparison tests based on the pumps, the test objectives, the pumps that are going to be tested, the engine that is going to be tested, the profile of past tests for that engine or pump. Sometimes we run a special test, like hydraulic lockup (A1 has been doing a lot of those). We don't do those often, so we go back and pick a hydraulic lockup test in the past, so we have a one-to-one comparison on the primary objective of the test.

Q: What other objectives would you consider?

A: For example, DCUA or CDUB halts. This is where you fail one controller channel. The controller switches to the backup channel and you see blips in the data. The pc momentarily takes a spike. We have a bunch of those, so if we ever do a DCUA halt, we'll compare it to one of the tests we've done in the past.

Sometimes you'll shut down from a lower power level (90%, etc). That has a little bit different trend on the shutdown. We'll compare the shutdown portion (we sometimes have different comparisons for start, etc). For start, when we first started testing the PW pumps, they wanted to start at 65%. So, we had to go way back in history to get some 65% starts to compare to. If you couldn't find it you would just compare it up to 65% and ignore where the one test went up and the other one stayed down. Same thing for shutdown. You try to pick a comparable test where you shut down from a comparable power level.

Or, for example, on TTB they're doing different CCV schedules during shutdown and start to try and smooth out some of the bumps and spikes in starts and shutdowns. Trying to make them less deteriorating to the turbine. You'd want to compare like CCV schedules so you'd compare to the last time you did a CCV schedule like that.

{End of side A}

{Beginning side B of Tape 5 Neely/Dave Foust}

That would be a pretty main objective for TTB so you'd try to compare to other tests with comparable CCV schedules.

Green run of h/ware is always a major objective, also CCV scheduling, hydraulic lockup, pneumatic shutdown, based on the engine's history - whether or not you have a prior run you can compare to, and starting and shutting down to a funny power level.

Q: During the start, you have see if there is anything different. Then you have to choose a test with a similar profile. That would be your main concern.

A: Engine starts are pretty common. If it was a normal start

A2-149

without a modified CCV or power level, then I'd pick the last test of that engine.

Q: So for start, you pick the last test of the engine unless there's an alternate CCV schedule, or you start to a different power level, and then you try to search on that.

A: Right.

Q: And you can tell that from the pretest?

A: Yes. The pretest has a thrust profile, except for the 1.5 second tests.

Q: During mainstage, then green run? Or you go to the last time the engine ran on that stand?

A: If you're green running h/ware, then you try and pick the last time that particular h/ware was run, if it's a similar profile. If not, then I'd go to the last time the engine was ran and compare it at least to the same engine. Then you'll just get pump to pump changes in your comparison.

Q: What if it's not a green run, then what would you compare mainstage to?

A: It would depend on the test objectives. If you're still in certification they'll have different profiles so you'll compare to the last similar profile of that engine. It's hard to compare a long duration test with a short duration test because you lose a lot of resolution in the data. You might want to go back 2 or 3 tests of the same engine to get a similar thrust profile of a similar length.

You want to at least match up a couple of power levels within the profile so you can say "ok, the turbine temps at 100% are here on this test and here on the other." They may not be exact profiles, but at least you get similar power levels and similar inlet conditions that you can compare and see that the turbine temps are running about the same as they were on a similar test. It's not necessarily an identical profile.

Q: Inlet conditions?

A: On green runs they are pretty standard. Inlet conditions are a secondary effect. They affect the lox and fuel turbine temps, which will trend depending on the lox inlet press. The fuel inlet press doesn't affect as much. The lox is heavier and when you pressurize and depressurize it, it will have an effect. If you cavitate a pump, you'll cavitate the HPOP typically. When they do a vent to try and simulate flight they're trying to see whether or not the lox pump is cavitating and things like that. In some tests they don't run vents. Primarily the power level will be the main effect, and vents will be a secondary effect. If you could line up 100% with the same vent conditions or 104%, or 109% with the same vents, fine; if not, you'd at least compare 109% to 109%, and say, "well this one was vented and this one wasn't." You know the vent causes some effect. Mostly the lox vent would be a consideration.

Q: If you were at 104% and then 109% on one test, and on the other test you didn't go from 104% to 109%, maybe you went from 100% to 109%. Can I compare the 109%'s and expect the same characteristics on the engine?

A: Roughly, depending on if the vent condition is dramatically different or something else has changed. That would be a good comparison change, and they don't have to be both at the same time.

Q: So, it doesn't matter if it's the same time in the profile, you're really looking at specifically power level.

A: Power level is primary, and inlet conditions second.

Q: And it doesn't matter what came before?

A: No, not usually.

Q: The data analysts don't shift thrust profiles thought, do they?

A: You can mentally, or you can on the plot program that we have. Typically, if the 100% portion on test A is here, on test B it's here, you're basically looking at a level. If you see that it's running at 1300 degrees here and 1350 here and we expect that because of venting, etc, then you can eyeball it. If it's way over here, then you can do a time shift to reference them together just for comparison purposes.

Q: Even on a green run, you'd like the pumps but only if the thrust profile is similar. If the profile is not similar, then you're willing to settle for the same engine. So, even in that case, thrust profile takes precedence over h/ware?

A: Well, if the main objective of the test was to green run a lox pump, my first priority would be to get the same build of the lox pump, if not then a prior build with a similar profile. If this is the first time this pump's ever been run then you don't have a choice.

Q: So, you want the last time the pump was ran with a similar profile, it doesn't have to be exact, just has to at least have the same power levels included.

A: Some way where you can at least get a close one-to-one comparison on performance. If you're looking for pump performance changes, it's hard to compare 100% at nominal vent vs. 109% at min vent. Trying to work the vent and power level out of it is harder. If you can get close power levels and close vent conditions to try and get a one-to-one comparison.

Q: So you also take venting or inlet conditions into account?

A: As a secondary effect.

Q: What about when you green run pumps, for example, do you have to pay attention to how you shutdown?

A: Typically, if you're doing a flight h/ware green run, you're not going to do anything that's not done on flight. Otto's very firm about that. There shouldn't be any strange shutdowns or starts.

Q: Is flight h/ware different from other h/ware?

A: Flight h/ware is built strictly to current flight configurations. It's typically very new, very good h/ware. There is a requirement that no flight h/ware exceeds 50% of the fleet leader. For example, if there's a fuel pump who's turbine end bearings have more time on them than any other fuel pump in the fleet (ground, flight, anywhere). They call that fleet leader. The bearing on a fuel pump that they're going to fly cannot have more than 50% of the fleet leader time on it. Including the RTLS abort (the longest abort scenario which is 753 seconds?). The current amount of time plus 753 seconds may not exceed 50% of the time on the fleet leader. So, it's newer h/ware and is typically the best that we have.

Q: What about h/ware that has already past 50% of the fleet leader?

A: It would probably go to development.

{More examples of things that have gone to the development program}

Q: Did those eventually become fleet leaders?

A: It might. Engine 2206, the engine that we just ran, had a

A3-151

more cracks in it than any that I've seen. It had something like 13 pounds of leaks.

Q: That was still labeled a green run, like an oxidizer turbopump? Wouldn't you have to dump a lot more lox through and have hotter turbine discharge temps?

A: I was talking to them this morning about that. It makes flight predictions almost impossible because you're trying to green run a pump on an engine that runs off nominal, and trying to see how that pump would run on a nominal engine. It's very hard. 2107 is in a lot better shape, but it has a problem in the discharge duct in the HPOP. There's a bad weld with a burr on the interior of a duct. There's a 4-sigma high HPOP discharge press. This makes everything on the LOX side run different. It jacks the disch press up, which makes the turbine temps go up, which makes valve pos's go up, makes speeds go up which changes the press going to the valves, which changes these valves here. Then you have to try and back out - "well, it ran this way on 2107, but the HPOP disch press was this high. On this flight it will run 100 psi lower, so what does that do to me." It's really hard to do that.

Q: If they went in and removed that burr, would that do anything for you?

A: It would help, but they'll never do it. That's a power head problem and it would take forever to fix that.

Q: How did you notice that the burr was there?

A: We noticed how high the HPOP discharge press was, like 3-sigma higher than the average. They went in and inspected it and looked at the weld xrays, and saw the burr.

Q: Which test stand has that 13 pounds/sec leak? Would that show up in all of the post-test summary sheets on the engine and instr problems section?

A: I doubt it. It would probably be in ...The combustion devices people would probably keep track of it (Don Charcl___). We have some of it because Bill Green and Tracy, and Brian also trace that stuff because it affects the performance of the whole engine. If you've increased lox flow to make up for fuel leakage, isp drops down - because you're not getting anymore thrust, but you're dumping in more fuel and lox. That was engine 2206. We've been green running pumps on that engine for about two years. When you run an engine like that for 2 years, hard, it will start deteriorating rapidly. There are a lot of recent tests on that engine where they have probably been tracking leakage. They could probably show you a chart showing the leakage.

Q: So you knew that every time you got 2206, it was leaking and you took that into account.

A: It ran on mixture ratio, they changed c2 to make it do that. But, the lox flow was increased, it affects everything. It runs mixture ratio but it's not really a nominal run. Anything recent on 2206 is guaranteed to be a little bit unusual.

Q: That should show up in the 2-sigma comparison when you compare engines without massive leaks like that, but not if you compared it to the last time that engine ran?

(Claudia and June discuss the comparator vs. the 2-sigma stuff)

A: It wouldn't (show up in comparison to the previous run on that engine) unless you had something significant change.

If you compare 2107 to just the last couple flights, the HPOP disch press's will be on top of one another. That's where we use 2-sigma. If you compare it to the rest of the fleet, in HPOP disch press, you notice that it's high. You can mask a problem that way. That's why we have start 2-sigma's and mainstage 2-sigma's. It gives us a better feel for how a normal engine would run and what

kind of variations you would see.

Q: Then you wouldn't you put the info from the ____ into your 2-sigma database, would you?

A: We have strict criteria on those. If an engine runs way off mixture ratio, or any of the phase2+ data into the 2-sigma. We're building a database strictly for the phase2+. Until that database gets big enough we'll compare it to our normal 2-sigma, but we we'll not include that data into the 2-sigma database. We also have some databases just for flight.

Q: If you green run a pump on a particular engine and then put it on another engine for flight, do you just visually inspect that pump after it comes off the green run engine?

A: They visually inspect, do torque checks, leak checks, etc. Typically after a green run, it's ready to fly, they don't run it again.

Q: Does it experience anything before green run?

A: It can, sometimes it doesn't. Sometimes when they get a new pump they'll do a lot of inspections when they build it. Then they'll green run. If it passes green run, they'll stick it on the flight, and not run it on any other engine.

Q: So it's never tested in any turbopump stand or anything before this?

A: Not usually.

{Dave pulls out a databook for a Pratt-Whitney pump and everyone shout's "let's not do this one"}

{End of conversation}

{End of 1st half of side B on Tape 5 Neely/Foust}

A3-155

Interview Session Date: 4/29/92
Darrel Gaddy

DG - He and the Martin people pretty much do the same job, he's just NASA. His main responsibility is flight.

Difference between the stands...

TTB A development stand and can't have long test on it because we don't have big propellant tanks. We have gone to 205 seconds duration.

A1 An open [atmospheric] stand and have long test capabilities, [because we can transfer propellants during the test. We have done 1000+ second duration test before.

A2 Where most of the flight engines are green run. We have a diffuse the stand to simulate conditions 80,000 ft up in the atmosphere, so we can throttle down. On TTB and A1 we can't throttle down because of flow separation in the nozzle will destroy the nozzle.

B1 Just like A2, it has a diffuser. Then we can go down to 65% power level. Fairly new stand.

When we had a new engine, we used to do 1.5 second test (a big burp of the engine) to do a system checkouts. We do not do that anymore. In looking at the data from all of the 1.5 second tests, only 2 engines out of 100 had any problems and even these problems would not have prevented the engine from going on to main stage. The next test is a calibration test (where you go in and make your software adjustments). The next test is a flight simulation (acceptance) test where we throttle down.

* Initial Test (not performed anymore) - 1.5 sec duration

* Calibration Test - 250 second

* Acceptance Test - 520 second

Test Case - B1-099 Engine 0213

Programmed 771 duration, but was cutoff at 1.66 secs. due to a failure OPB. We have purges that come into the preburner domes that flush out through the turbines and just goes out the nozzle. [During] prestart the purge is nitrogen and [during] post shutdown we run helium through the engine. There is check valves in the lines, so you can flow into it once you have your nitrogen or helium up to pressure, but once your engine is started you don't want hot gases flowing back out of the purge lines. There was a check valve which leaked during this test firing. There is a pressure transducer upstream of the check valve. The check valve is monitored at engine start to 2.3 seconds, so if the check valve is going to leak, it will do so by 2.3 seconds. After that we don't look at the measurement.

This is what we did. We suspected a leaky check valve, based on that the purge pressure follows MCC PC trace. That means we have an OPB purge, FPB purge and MCC purge all at the same time and they each have their check valves. And we have sensors for OPB purge and FPB purge and not on MCC purge, but if you have a leak on the MCC it can work its way back up and show up on OPB pressure transducer.

{ Goes to plots from package }

These [parameters] are plotted from engine start to 6 secs. The other two tests are 1.5 sec tests that we did on flight engines. The shut down [1.66 sec] was so close to 1.5 sec. that we went ahead and compared it to 1.5 second tests. FPB shutdown purge pressure (PID 148) and OPB shutdown purge pressure [PID 149] from time zero to 2 seconds. Both pressures should stay down at atmospheric pressure for the whole test, but they started going up. The redline is set at 50 psi. The redline cut here (pointing at plot where values exceeded 50 psi) and then the engine shutdown. We did "tap data" here showing when data went above the redline.

{ Goes to the pneumatic schematic }

These are the check valves we are talking about here. This is your OPB purge and your FPB purge. And we have these orifices here to control the flowrate, once we do apply the pneumatics to here. And then we have these check valves here.

These are the things that can leak. This (he is pointing out blocks on the schematic labeled "pressure transducer") is the pressure transducer that relates back to this measurement, OPB shutdown purge pressure and this pressure transducer here [is] PID 149 for the fuel.

But on this one [test case] you'd think that if you'd had an OPB purge pressure cut off here exceeding, you'd have this check valve or this check valve leak (he is pointing to the two check valves downstream of the OPB pressure transducer going to the oxidizer side of the power head) coming up to get to this pressure transducer. But on this one [test case] we had a main chamber oxidizer dome purge

e [check valve] leak and it came back, work its way around this way {tracing the path on the schematic to the pressure transducers} to pressurize this transducer.

Question: So really you could have any one of those check valves leak and you would see it in any one [of the pressure transducers], so you really couldn't distinguish or could you distinguish which check valve, besides with a visual inspection?

DG - That's the best way, after the test they go through and leak check them and they will leak, but sometimes they don't do that. We had one of these check valves leak on a stand. We thought it was the preburner check valve. We went in and replaced that check valve and in the next test the same thing happened again. And so the way we are looking to see which check valve leaked, is we're taking this trace [the trace of the shutdown purge pressures] and comparing it to either the MCC pressure, FPB pressure or OPB pressure [which] can drive this pressure [shutdown purge pressure]. And so we're looking at when these pressures [shutdown purge pressure] increase, if they correlate good with these other pressure increases. And that's the best way, to go back to the pneumatic schematic to see which check valve is actually leaking is to see which one of these pressures [preburner or main chamber] increases at the same time this one [shutdown purge pressures] is [increasing].

Question: Which is what they meant by "follows the MCC trace"?

DG - Yes

Question: "Follows" didn't have anything to do with time by how well the graphs meshed?

DG - [Meshed] with time.

For with the check valve leaks, what we're trying to do is, if I was going to look at the data and [wanted to] tell which one of the valves leaked, I would compare the actual trace of the one that cut me [caused the engine cutoff] to each one of these [the three chamber pressures, OPB, FPB and MCC] and see which one it correlates to. Then I would say that check valve leaked. It [is] really hard to say that too because we have this pressure transducer over here called OPB purge pressure transducer, but we have this one leg which comes over here and looks at the fuel preburner. So we only have this one [pointing at the fuel pressure transducer] that looks at this check valve, but this one [the oxidizer pressure transducer] goes to both of these two [pneumatic legs to both the fuel and oxidizer preburner oxidizer domes], so you could have a fuel preburner check valve that [effects] oxidizer [shutdown purge pressure]. It gets kind of messy just trying to find out which check valve actually leaks. And you could have this one [MCC oxidizer dome check valve leak] coming in the back door and get that one [oxidizer shutdown purge pressure]. It takes awhile to look at these things to tell which one, [and even then] you're really not sure. And then so the best thing is to go back and inspect the engine by doing leak checks and you can tell [for sure] usually by the leak checks.

So that's what happened. We had this check valve [MCC oxidizer dome] leaked and we started building up chamber pressure here [in the MCC chamber] and it worked its way back over and down this way [he is again following through the schematic path] and we had to cut [engine shutdown] and we had to cut on OPB [shutdown purge] pressure transducer.

Question: Do you have to look upstream of the pressure transducer that is reading anomalies?

DG - Yes anything that will feed into it.

Question: These chambers start [their respective pressure climbs] at [different] times too don't they?

DG - Yes, they do. First we want to start the engine fuel rich. That way [since] LOX is really nasty stuff, you get a LOX rich [start, and] the engine is just melted. So what we do is we start the fuel side first. That way we just flush the whole engine with hydrogen and we have a cooler start. We prime the fuel pump [actually the igniter faceplate is primed] first get it rolling and they we prime the main chamber and then we prime the OPB.

The way we [define] prime times is... In the fuel side we are looking at the fuel pump speed, so we can actually see a slope change. The fuel pump will start to

urning just from being pressure feed from the facility, then when your injector ignites it will start spinning that pump faster and there is a slope change in that trace. And that is what we look for [to determine when] the fuel prime [occurs,] is that slope change.

Question: Do you know what that slope change is?

DG - We've got a program that calculates it.

Question: This [MCC prime time] is was under 200 psi right?

DG - MCC [prime time] is when you get above 100 psi in the chamber.

Question: And this [OPB prime time] is actually the slope changes from negative to positive?

DG - On LOX turbine temps, because we don't have a speed on the LOX pump, so whenever your turbine temps go from negative to nonnegative slope we say the igniter faceplate is lit. If you are going to do a prime module or logic, I'd recommend [that since] we have already coded it and you take what code we have, because we already have a database based on that one and I'd like to keep it the same, so it's repeatable from running on the PE4.

Question: Does that code include all controller phases?

DG - It does a lot of them. That's their ENGSYS (ENGINE SYSTEM) [program]. It's written in C on the SUN station, but we have the same code on the PE4 [written] in FORTRAN.

Prime time for the main chamber is here. That's about 1.4 seconds. And that's just when it goes above 100 [psi]. That one's real easy to calculate. Fuel pump is the slope change [looking at the startup fuel pump speed trace]. It is really hard to make it out, but can you make out the slope change right there. It's rolling up at this slope and then it changes over to that steeper slope. That's what we call the [FPB] prime time. And then for the LOX turbine temps [looking at the LOX turbine temps startup trace], this one we would say it primed when it started to turn around there, that's where it primed. It would be the latter of the two channels, so you'd have to have both of them turned around to get a prime time [DG not real sure about that].

What I'd like to see if we have this shutdown from this OPB purge pressure, is that you would know that we had this shutdown due to a violation of this redline and then it would come back and tell me that that most probably it is a check valve on the OPB side or FPB side or it follows the trace of the MCC PC. And that way we can go and ask for an inspection on those check valves to try to isolate them.

Response: That's why I was trying to find out how they figured out it was the MCC check valve or whichever check valve it was, how they were distinguishing that. That's what we're trying to flush out in these case analysis.

DG - I think what we did by hand, we made a plot where we had the preburner [shutdown purge pressure] trace and we followed it on the same plot with MCC PC to see how close they were and they looked really relatable and scalable. But if you look at the FPB [chamber pressure] versus this one [the shutdown purge pressure] it didn't look the same. The pressure [shutdown purge pressure] did not start rising at the same time.

Question: And you looked for something that the pressure started rising a little bit before, as opposed to the FPB..?

DG - No. It would be the same time if it were. If it were the MCC PC, the pressure rise, whenever this pressure increased, the purge pressure would increase.

Response: Simultaneously

DG - Yeah

We have two different things here. This one's [the anomaly summary] saying that it was the MCC PC trace and this one's [the post-test review] saying that it was an OPB purge line check valve. I think that this was the test where we had them back to back. We thought it was an OPB purge line check valve because it was a violation of the OPB purge and then we tested the next test and it did the same

thing after we replaced the check and we then looked at the MCC trace.

Response: It says there [on the anomaly summary] on the action item that it did replace the OPB purge line [check valve].

DG - I think these were back to back tests and this was whenever we started looking at the traces to see which one it is. This one [post-test review] says its an OPB purge [check valve failure], but I don't believe it, I think its MCC [check valve failure].

I said something about PIDs, are you familiar with PIDs? MSIDs?

Response: Yes, PIDS basically for CADs are 1 to 301 and anything greater is facility.

DG - MSIDs are CADs. Are another way of calling a CADs PID except its [a] flight orientation. We used to, we recently fixed it, on flight data all we have is CADs, we have no facility measurements, so everything under 300 we had, but we couldn't use PIDs we had MSIDs, which are E41P{its engine number}027D. Recently we switched that over, so now we do have PIDs on flight data also, so you can just work with PIDs, instead of MSIDs.

Questions: Do the data files now contain MSIDs?

DG - Yes, they contain both of them so you can use either one. I think its on the header on the file, which cross references the PID and the MSID to the same whatever in the database.

Response: I know they have the MSFC format to work with now. And it seems to have more header information and I remember there was an extra column there and I couldn't remember exactly what it was telling me, but maybe it was the MSIDs.

DG - Yeah, it was not all flights, it's the last flight and the FRF that have that [PID information] in it, but everything before you have to use MSIDs.

Response: OK, so if you were looking at the test data file, you wouldn't find the MSIDs in the header?

DG - You would. It would be assumed it was engine 1 MSID for that PID ground test. But you can [use] PIDs, [so] why waste [time using] MSIDs.

Questions: The oxidizer preburner purge line is between the MCC and OPB right?

DG - How this works ... This is the PCA [pneumatic control assembly] on the engine. [It can do the following control functions] it can turn all the servo valve and pressure operator valves for the pneumatic assembly and it bolts along the side of the engine. Bring helium in here and the if we were going to purge the OPB or FPB and we do this after shutdown, like at shutdown plus so many seconds we turn on the purge. And we have helium here and it runs over to this shutdown purge solenoid valve and comes up to the valve and stops. And then whenever we energize this solenoid, it allows it to go in-the-in and out-the-out and it comes down here to PAV4 and PAV5 and it goes in the c-port which is the control port. And it allows this 750 psi helium to compress this spring and that allows the helium which comes in this way to go through in and out-the-out and to go through the check valve and out that way. So we control it this way it goes to the control port and we flow that way.

Question: Why couldn't [this failure] have been detected before start?

DG - You mean the leak? Because we had no pressure in the preburner. As the preburner pressure increased, the leak went back this way. We did a leak check before the test and it did not leak, but during the test [firing] it leaked.

Question: Does the check valve prevent the flow from going in the reverse direction?

DG - Yeah. It allows the helium to flow this way, but no flow the opposite way.

Question: And then in post-test is the flow [check-out for the check valves] done every test.

DG - No we don't do it for every test, but we had a problem so we looked at it.

[End Of Tape]

DG - [Discussing the post-test information in the package]
...So we were green running the duct in the MFV actuator. We're doing a certification of a LOX pump which is trying to get this design of the flight time on it so we can fly it. All this junk on there is what we were doing.

And then we have some valves and some configuration on the facility that effects engine performance. Fuel repress flow control valve, on flight we take part of the hydrogen flow that comes through the pump that goes through the MCC [coolant circuit], that [then] goes to drive the LPFT. At its exhaust, we tap off here and then in flight we take this and stick it back in the ET [external tank] to pressurize the ET. We can have a flowrate here (pointing to the schematic) of 1.3 lbs/sec maximum or 0.2 lbs/sec minimum. We are going to go from max to min at 300 seconds. And this effects the LPFT power and so it will so it will effect the speed on the pump end, it will increase the energy to that pump. And so we'll see that in the data...

A3-159

User Interfaces with Erik Sander

4/30/92

Disclaimer: It may look like direct quotes, but it's not - exactly.

(tape: 000)

Erik: On the output, I would like to see it broken up into three categories. The first would be what I, the system, consider an anomaly. The second is what I consider instrumentation.

Q: Bad instrumentation anomaly?

A: No, just bad instrumentation. You can call it instrumentation anomaly, whatever. I've seen an event, I've determined it's not in the engine it's just a piece of instrumentation that's gone south on us.

And the last is what I call observations. They're real phenomena going on in the engine but it's something maybe we've seen in the past or something like everything reacting to a slow start. That would be under observations. My turbine temps come up slow because I have a slow start. My lox pump speed comes up slow because of my turbine temps. That's under observations. We have a lot of those.

You'll find that as we go through these tests, ... you get very few real anom. Maybe one or two a test. In terms of observations, you can get 20 per test. In our mind we quickly delineate what's an observation - what's an anomaly based mostly on history. We've seen this before ... analyzed it ... figured it out ... not a problem with the engine therefore it's an observation. A lot of times in the data review we'll point those out. That's one of the first things I'd like to see on the user interface (UI).

Q: How do you like to see that ... do want to see three boxes? ... Three buttons in case you see those things you can click on them?

A: No, I'd like to see one button because I want one block that says, "Here's our observations."

Q: One button for observations, one button for ...

A: No, one button for all three of them ... one block. I'd like to see them ordered within that block. The top would be anomalies, the second would be observations, the third would be instrumentation. For instance, if I have two anomalies in a test ... I want one big box on the screen and the first two lines being, "Here's anomaly #1, here's anomaly #2."

Q: ... Kind of like her (present user interface's) observations screens.

A: Yes.

Q: Up at the top? You like that?

A: For ex., lets use this test right here (683)... it's (the UI) thinking - could you give us something that tells us it's thinking? ... (choosing another test that has postulates) ... Click on the box, that's not bad ... This is the kind of thing I'm talking about.

Q: You like that.

A: I like the box that says, "Here's what I picked out."

Q: It seems that if you have three like this right next to each

A3-161

other, that you would know immediately to go to the one and scroll up or down.

A: The problem with that is that it cuts it down. The other thing I want is wrap-around.

Q: Would you like some kind of indication saying what's an anomaly ...?

A: Yes, write "A" for anomaly. PUT the first two anomalies up ... anomaly, anomaly then "O" for observation, "I" for instrumentation.

Q: What about a break between - like a line between areas ...

A: That's fine. Anything like that just as long as I know what's declared an anomaly, what's declared an observation, what's declared instrumentation. And I do want them all, if I can ... I want to do as little scrolling as possible. I want it put up there and I go, "oh, OK", because sometimes I'll look at one and relate it to another.

You could call it "double checking." YOU could have an anomaly here but here's an observation that's feeding it.

Q: But you have an explanation of that. You don't want it to just sit there and say, "we have a slow start.", you want it to say, "we have a slow start because of this and this and it's confirmed with this and this and this."

A: That's right. I want as much explanation as possible as to why it came to that conclusion, without getting ridiculous. What I want it to give me is, "Here's what I think, here's the parameters I looked at, and here's why I think that thing." If there's some histories behind it then here's the histories.

Q: You don't expect though that the bad instrumentation and observations and the actual engine anomalies start to be able to confirm one another.

A: No. They shouldn't. ... I just want to make sure that the program isn't feeding into itself - that it says, "I've got this anomaly based on this, this and this" and later on I see a piece of instrumentation that's bad, that it's using something like that.

Q: What do you think about ... how do I get back?

A: ANother point. There's several things that you've got to close this thing to get back to this one to get around to that one. I would like it so that you can get around to the different sections as quickly as possible without going through a bunch of closes. In other words, a network instead of a hierarchal type of thing.

Q: So right now we should be able to click on this back here and bring it up.

A: Yes. This guys up front and I want to go over here and ... just click on it ... I'd rather just click on that and keep this one up in the background ... keep the close button in case you want to lose it. ...

I'd like a screen like this ... to be pretty sizable.

Q: If you want the idea that you can click on it, click on this, you can't make the screen too big, but it can be bigger.

A: OK. Titles up here would be nice because then you can see which screen is what. ... you could increase the size out here and leave the titles up here so ... you can see the titles and ... bounce them on top

Q: The only problem is that if you bring this one on top ...

A: Change the sizes ... if you can do it.

Q: ... could we prioritize that?

A: yes, give it lowest priority.

Q: Can you take it where you would click on an icon ... where you double click to bring it up ... to go back ... just click it.

A: That's fine. All that stuff is lowest priority. ... The highest priority to me is quickly seeing everything this thing has to tell me. Probably the second highest is to bounce around w/o closing three screens to get to another one.

Q: Now, if we go back to here (main 'ds' screen) ... What about this? ... Do you like the screen coming up so there's something anomalous it splits it per ... LRU.

A: Yes, but you'll have to have something says 'SYSTEM' or something like that. Right now, this is an intermediate step because anything wrong is going to be in this box. ... If anything, take off the words, ... we've dealt with the HPOTP, LPOTP (to know where they are) ... again it's a low priority.

(Tape: 100)

Q: What about this (losing titles across the top as scrolling down) I thought it was a problem ... however it only goes to this one (the latest test received).

A: I'd rather ... if it gave me an extra test lose it, but it doesn't do anything extra for me. ... give me more room to see tests. ... All this is wasted area to me. it looks nice, but I'd rather be able to see 4-5 more tests at a given time.

Q: I almost wonder though, you see the way this goes across to tell me it's complete, that's only for your current test. So that's all you need to know and everything else is completed. So, could we have a window which just shows a list of tests.

A: You could put down test, test, test, test, something like this.

Q: Maybe a pulldown menu which lists all the tests. Or keep this test line just like it is, and then put a bar there, ... a separate menu, and then put all the tests there.

A: How about above the bar, put any tests that don't have checks in all the boxes. So you know what's running or what isn't complete or what you can't run right off the bat.

What I'd like to see is all the tests ordered in terms of ... here's all the 'A1' tests, here's all the 'B1' tests, here's all the 'A2' tests.

Q: How about a button: 'A1', 'B1', 'A2' and you just click on it and it gives you a popup or pulldown menu with all the tests listed.

A: That's fine, because then we're searching for an A1 test and we're plowing through a bunch of these.

Q: How important is the option of not actually going through the list and searching, but (instead) typing in what you want.

(tape: 125)

A: Either one. As long as I've got a list telling me what's available. ... Have it pull up a list and put it numerically, 625, 624, that way it's easy to find.

Q: Reverse order?

A: Yes, with most current first. ... a box A1, B1, A2, TTB.

Q: You want to get to the plot package easily.

A3-163

A: Yes.

Q: Do you want it to be a part of this screen, or to be on a third button pulldown, like this middle button which comes up with certain things. Do you want the plot package under here (middle button) or under "options" (on 'ds' main window)?

A: ... put it on the options menu. If you can put the plot package there. Put access to databases; what I understood was that you could run everything from this screen once it's done. ...

I envision that one day I could sit at my desk and be able to run everything from this program. That's why I want plot, databases, history bases whatever.

Q: Is it set up so that the person in the middle of the night can actually enter the unit numbers? How are we going to do that. How are we going to get the information we need? Right now we have to run 'new_data' to get that info in.

A: That's probably how it's going to have to be for a while. We don't have any fast way to do that. ... It is automated in one system, 'TRACER', it's a Rocketdyne system, but we don't have access to it.

Q: Right, and I got the impression yesterday from talking to Mark Neely, that that's just a hard core (??) tracking system so they don't update it immediately, and if they don't update it immediately, or before the test is ran, it doesn't help us.

A: We can get Boeing people to do that.

Q: Do you want (new_data) to be separate?

A: Yes. ... bring up the screen, enter the data, hit 'go' and you're done. ... With that, we ought to have a check so that when they hit go, it tells them it's going.

Q: Do you want a screen to stay up and say, "Loading Data", and when it's done, say, "Finished", in case there's a problem, then they'll have to restart it.

A: Yes. Or have it tell us first thing in the morning that there was a problem, or whatever.

Q: Do we want them to be able to enter multiple tests?

A: Yes. What I want is, you can enter these things, hit the go button and then you can do that connectively or whatever, you can't do two test now at the same time from what I understand anyway, you wait for one to finish then you kick off the other one. ... I would like to enter the info for the two of them and that way when it finishes on the first one you can kick off the other one. And then another screen which says so much completed or whatever ... pretty much what you have here.

Q: This stuff you need under new_data (first 4-6 ? columns of main 'ds' screen) this (last 1-2 columns) belongs under this ('ds').

A: The only thing I care about on this screen is the end block. I assume if I have a check there, all these other ones are done. This is nifty since it tells me where we are.

Q: But this stuff is dbload stuff ... this stuff is for the diagnostic system, so leave this and maybe split or something like that. Like you said, you don't care, if you got the care, you've got that.

A: When you get a system you're going to have to replace these boxes with ones that say system.

If we get into this (selecting an LRU) I like the buttons that have all the PIDS on them. ... What I would like is the option to click on multiple PIDS for plotting.

Q: but if you scroll down (the plot screens), you get the other ones that are important ...

A: that's for an observation. ... If I double click on that, then bring up that one PID, if not I want the ability to go click, click, click and the last time, I double click.

Q: Like a 'go' button.

A: No, it's just that last one I want, I double click on. If I just want 234, I double click. If I want 233 and 234 I go click, double-click. Double clicking just means " I'm done putting in data".

Q: If MOTIF won't allow clicking/double-clicking objects to select, could you make a go button, and click on that.

A: That would be second choice though. ... the other thing is that we'll plugged into PV-Wave or some plotting package so I could go off and plot on that. But this is convenient because here's my observations, here's my picture, while I've got the observations up there I can look at what we're talking about.

Q: I've heard that we want hard copy plots.

A: Yes.

Q: directly from it.

A: Yes. If I go in here and click on that, whatever and I find exactly whatever it is I want, ... I want to be able to go 'clunk' and get a hard copy.

(tape: 220)

Q: ... you looking for one time period to plot all the PIDS against?

A: I talked about this with Pam. She basically set this up from some time scale 0 to whatever. I want the ability, and I think she did it here, ... If I rescale this from there to there, and I decide this is the area I really want to look at, I want to be able to just hit the button and globally rescale the whole thing, so that any plot I look at from now on will only be in that time frame.

That's important to us because what we'll do is go in and say, "I wonder what the heck happened here?", and we'll want to plot data from this time point to this time point. The way her system was set up, I'll plot that time frame of data, but when I go up to another plot, to maybe investigate what's causing this, I've got that original time scale again and I've got to go in there and shift my time scale. That's a very important thing to us because it saves a lot of time.

Q: Do you want that global rescale if you had a plot?

A: No, give me the plots the way they are now. ... She set it up so that if you want it from here to here, I hit global rescale.

Q: SHE set it up to global rescale that you have to go and type it in. Is that OK.

A: I've got to tell the machine at some point I'm looking at this time frame to this time frame.

Q: But it seems you want to go in and click hit that, and then it does it?

A: Right.

Q: That would be your druthers.

A3-165

A: Right.

Q: Right now it's set up that you say, "I want global rescale", and I want it from ... you have to start filling out this seconds. ... then you've got to turn it on ... then close ... and replot.

A: I would rather just go, "look here and here."

... It's easier to me if I could go click, click. NOW in that, how about an option, - I've got several buttons when I do that. The default is I want exactly the times that I specified with that pointer, but then give me an option to give me the times to the nearest second or the nearest five seconds, the reason I say that is if you do this, you may have 100.374 (from the pointer) to 260.32

Q: ... looking at this plot here, you can't really pick up seconds anyway, so do you want it to go to the nearest second (tape: 262)

A: You look at a plot like this there are some anomalies in something like a ten second time frame, I want to look at 10.5 or 10.3.

But on something like this, I'm just trying to get a rough estimate from there to there in terms of time. So have it round to the nearest second or five seconds or whatever.

Q: So perhaps on a full plot, you round off to a second

A: yes

Q: And as you start zeroing in you start wanting to go to tenths of seconds.

A: Exactly. So have the default that if I click here and here it rounds off, for me, to the nearest second. But then have a button out here that says I'll actually want those particular times because when I get to the small ones, it's not going to round off to the nearest second.

Q: What's considered "small". A time scale under what?

A: Let me worry about that by just putting up the button that says "round to the nearest second? Yes or no?"

Q: OK, it's click, click, pop a button, round second, round what? and maybe you'd have a button under "round?".

A: I would put it as "Don't round" because then as the default it's going to round to the nearest second. That's what I'm usually going to want to do. That way, when I exactly want these points, I just hit that button and it doesn't round.

(tape: 279)

Q: I'm with you. So you click, click, automatically pop up a button ...

A: Right, I go clunk, clunk and a button pops up and it says, "Don't Round to the nearest second?". Now if I leave that alone and I hit plot, then it goes, "Erik picked 100.34 so therefore I'm going to round to 100." If I hit that button, it says, " Erik picked 100.34 so I'm going to 100.34."

Q: Do you always want it rounded down or rounded up ...?

A: Closest second.

Now, while I'm on that, and this is a high priority one, don't give me goofy scales.

Q: What do you mean by "goofy scales"?

A: Goofy scales is like 17 seconds per tick... Some programs say ... I'm going to have 10 tick marks per plot and all it does is take the endpoints and divide by ten and I get 5.37 secs. per tick. That makes life tough for us. Invariably what happens is you're looking at a plot and you start looking "I wonder where this point is?", and you go down here and what it looks like is 4/10 ths of the way from there to there. That's easy to do an a second but impossible to do on 5.37 seconds.

Q: So it should be multiples of 5, 10 ..?

A: The point is to vary the number of tick marks on the plot so that it gives me a reasonable scale on the X and the Y axis. The Y has seen the same thing. It'll start at 1137.63 and it'll end up at 1547.64. You got ten tick marks in between and you're sitting there with your calculator trying to figure out how level this thing is.

Q: Is it all right that you have more graph here than data. So instead of this scale being 600 could it go to like 650 even though it (data) stops at 600?

A: I'd rather not and the reason is typically what I've been seeing is you get the same size graph and you just compress you're data more. I'd rather you just change the number of tick marks to get something that's reasonable. In fact, ... I've seen a lot of programs that instead of rounding off to the nearest second rounded off to the nearest two seconds to get a reasonable # tick marks and reasonable division between ticks.

Q: That may be something we just end up playing with and then send it to you if you're happy.

A: That's fine. If you want to round off to the nearest two seconds, that's fine, or three seconds, whatever. Just give me a reasonable number of tick marks. That's anywhere between 8 and 15. Put the tick marks so you get a reasonable interval between the tick marks and that's something I can divide by easily.

Q: Would it help if we put grids on it or would that make it too busy.

A: A lot of plots that helps. You can do major grids if you want. Again that could be something we can play with. It should be easy for you to change. I would start out with major grids.

Q: How about labels? Size?

A: I need them bigger than that.

Q: How about if that means a smaller graph?

A: That's fine. The reason for having the bigger labels is that we have to turn these into view graphs that management has to see on the board.

Q: How small of a graph is acceptable on the screen.

A: On a typical paper you want to fill up most of the paper with the graph. By the way, we like rectangular graphs. It's a cultural thing ... Put the labels on top because it's the easiest way to show me where we are. I don't care about logos on the plots. I don't care about the time the plot was made; I don't care about any of that stuff. If you want, you can throw the date the plot was made down here.

Q: What about the test number, not at the bottom. You want it at the top?

A: I like the way the PE does it right now, where for a single test, it puts it at the bottom left. If it's a multiple test, ... puts the test number and then the PID.

Q: Well, if you have a key over here, wouldn't you just put the

A3-167

PID and test separate?

A: I would rather have the rectangularness and I would lose this area over here.

Q: You would put it at the top.

A: Yes.

Q: But you would still have the line type?

A: Yes. ... If I could have dashes and lines and stuff like that ...

Q: AND different colors?

A: Yes.

Q: That would only help on here (screen) but not the plots.

It wouldn't help on the plots, but you would have different line types on the plots.

A: Yes, Give me the different colors because of analysis. You won't be able to use it in the presentation, because even if we have a color printer, we don't have a color copier, but it helps a lot on analysis. I can see much better with line colors than anything else. Give me line types if you can. Give me symbols if you can. Lose these bars.

Q: You don't like those? Why not.

A: Because I want to see every data point.

Q: But you have a good idea if you see the error get really big then you know you really want to see a data point. If the error bars are about the same size, then you don't want to see it.

A: I want to see every data point. You're asking what I want.

Q: Can I have it as an option?

A: Yes, but make the default every data point.

Q: Fine, I'll compromise.

(tape: 360)

A: ... Keep in mind that if you have one spike, it won't change your sigma that much. You can have a horrendous spike out of 50 data points...

Q: If you have a horrendous spike you will. If you have a tiny spike it won't.

A: To us, 20 psi on mcc pressure is horrendous.

Q: what is it's variation normally?

A: 2-3 psi. ... (much more discussion, incl. resolution. Bottom line, full data plots with option for 1 sec/avg with error bars.)

... What I would do, is that whichever mode I switch to, keep it there. So if you switch to error bars, keep that on till whenever you switch it off again.

Q: A global thing.

A: yes. A lot of the stuff we do is really global. Like the time spread. You're only interested in looking at this one thing now so you want that globally. ... A lot of times it's easier ... if you see them the same way....

Identify the parameter on top - that's fine.

Q: give reasonable suggestions as to what you want when there's more than one graph. We have some limitations.

A: One thing I like is the multiscale fcn. we have now on the PE. That means on one plot, you can build multiple scales to put multiple data on there. e.g. lox turb. temp. a pc and something else that have vastly diff. scales on the same plot.

The reason we use that is allows us in one plot to look at exactly when things are changing. That's what it's really used for.

Q: like this one.

A: Yes, though this one is confusing because it has too much data on there.

(tape: 420)

Q: Can you shadow things. ... when you have more than one graph and hold it up to the light ... can you do something similar on the plot. Would that be reasonable? Would you ever want to see one right behind the other?

A: That's good for analysis, the problem is when you want to present this stuff. We'll spend a lot of time analyzing then we'll get the final plot that says, 'heh, here's the answer', what you want to do is punch the print button. Then you could take it off to your boss and say, "Heh, here's the answer in one plot."

Almost anything that we look at, we want the ability to print. ... with the exception of the up-front figures and schematics. ANY data, any plots we want to look at because a lot of times you plow through the data and you finally get to that one thing that you say, "Oh my God, this changed and therefore caused this." You've got it in front of you and you don't have time to get your boss to look at the screen. You've got to get a plot, FAX it off to Calif. something like that.

Q: What happens when you have ... the option that you have a scroll bar to view different graphs... do you want 4 graphs at once on a page?

A: The problem is you lose the accuracy. Maybe one graph here and a little thing down here that tells you here's your other graphs that it's already pulled up or whatever. You're talking about the anomalies I assume .. click on an anom. and it says here's a graph.

Q: How about the one where you click on several PIDS and double click the last one. Now you have 4 pids you want to see.

A: On one graph.

Q: You really want to see them on one graph.

A: Yes, we normally won't put four on unless it's something like prestart because the scales get horrendous.

Q: The way the individual detailed graphs are being brought up are by LRUs. Are any of these graphs, where you want multiple pids on, across LRUs?

A: yes.

Q: How are you going to do this.

A: You're going to have to have a system block which will list all the pids on it - not all the pids but I could give you a good defn. of all the pids that we look at.

Q: how do you want those listed? in a pulldown.

A: I like the way she's got them with the schematic.

A3-169

Q: So we have to have a first level schematic with all of them up there.

A: yes. just an engine schematic with all the pids up there.
(looking at first ds screen.)

Q: (garbled) ... huge?

A: just shrink it down. we know what the pids are and where they are.

Many times, we'll pull up pids from over here and compare it to pids from there. Again, I understand we're going to have a plotting program dedicated to plotting what we want. If we can't do it up here, just give me the option to hit that button and use the plot program right away. I don't want to duplicate it ... on here.

ON this one, if we could use this engine schem. format to just run pv-wave, then do that. IF not, if you could only give me one pid per plot, that's fine too because I still have the tool in front of me to do multiple pids and mult. scales.

Q: Coloring things all right.

A: Red = anomaly; yellow = observation;

Q: what's instrumentation

A: mauve - I don't know.

Q: She has this thing where she doesn't list some things, like observations. Do want everything listed and forget priorities.

A: She's got some things in her observations database that say, "these two are ok. ... not using it for further reasoning". But if something is determined by the system to be OK, I don't want to know about it. ... I'm going to assume that anything not flagged by the system is OK

(tape: 504)

In terms of listing things by priority, that's ok, if you get a chance, but I wouldn't put a whole lot of effort into because usually there are only two or three anomalies in a test, and in our mind we will quickly know what we consider a high priority.
...

I like the ability to click on an anomaly and pull up the two or three parameters.

Q: ok. Here's one. I click on LRU and up comes the schem. - no test number - now I'm lost. no importance to you.

A: If you have some extra room, put the test number up there. Typically if we're going through an analysis like this, we know it.
...

Q: Before we were talking about pulling up several pids from different LRUS, how about other tests?

A: Again, we need to talk about where we're going to over-run the plot program. If you don't you're going to do a lot of work they've already done or have plans to do.

What I would really like if for this to give me the history where this pump has been run before. That's where I talked about options before. One being database, or hardware database, being able to access the database quickly and say I've got this fuel pump on here and it's run on these previous tests. and tell me which tests.

(tape: 536)

Q: Do you want to see the hw unit #s that are on this test?

A: Yes. and where it's run before. We've got that database already. We upkeep that.

Q: I thought you only did that for flight.

A: No. we do that for all tests. We've got that database together. It's only a matter of getting it over to Ingres or something and for you guys it means having an option that says I can go off and search it and find out what previous tests I've got and then at the same time in the options I've got aplot so I can go off and get the plot program to pull up that test

Q: Well most often though, there's going to be that one part which tells us which comparison tests you would normally compare to based on (what D. Foust gave us for criteria.) The normal one you would pick would automatically be there. If you wanted something different, then that's what we would have to coordinate with Jeff.

Erik: When you say it'll be there, when I pull up the plots, will it be on the plots?

June: I would imagine so.

Erik: Could I have an option to take that off?

Q: you don't want that there?

A: Sometimes I do, sometimes I don't. There's a lot of times I want it there because I want to see how it compares with whatever, but then I'll see a specific anomaly at a time zone in this test that I know wasn't on the other test.

Q: Ok. Another Global thing. ... A show/not show option.

A: yes. using a default to show the comparison tests is a good idea but give me the option to take it off because I see an anomaly on this test, I know ... it's not there (on the other test) and our graphs are a lot cleaner from that point on.

Again you'll have to talk to ... Jeff or Jean to make sure we're not over-running what they're doing or have done. I just want the ability in one place.

The thing I want to stress is ... I would like to have more working room.

Q: Can you think of anything else.

A: I like the clicking and pointing. throw as much of that as possible.

In terms of the plots, ... You know how you could pull up one plot, can I have ... the ability to have another little window in here (on the major plot.) and have a plot of something else? ... PC or whatever from that test?

Q: Why?

A: A lot of times we're showing perturbations of the turbine temps. The first thing that everyone asks is, "what are the power level changes?" So you would want a little guy in there that says what the pwr. lvl. chgs. were like.

Q: Why wouldn't that be the same thing as a double scale and plot it?

A: Because for presentation purposes ... it's clearer. Again, low priority. If you get it fine. if not, fine.

Q: What about if the Y axis on this one was this big (about 1/2) and you put the PC underneath it.

A: No, I've seen those & I'd rather ...

A3-171

Q: If you can't do it (with the little screen) don't do it at all.

A: Right. If you do it that way, all you do is squash the data down.

Q: I don't think that's a pv-wave command, but I don't see why you couldn't write that in C and write it to another viewport and open it up.

A: Again low priority.

That's about it. Again the colors are pretty important, especially when we're analyzing. You may not be able to print out the colors on plots but to us they're important because sometimes the data's right on top of each other and you're going which one that is ... different line types will help too ...

Q: Different line types will help too when you pull it off.

(tape: 615)

A: on scales like this you can't tell line types apart and if you have a little bump here, even if you have diff. line types, you can't tell.

Q: Ok. when you press on the pid by itself, ... you want to show the full plot 0 to cut off or whatever. How do you want that defined.

A: You have to default to something. I would default from 5 to cut.

Q: Always default from 5 to cut?

A: Yes.

Q: How are you going to want to handle ... ????

A: I guess you're going to have an option to enter the time. We're going to always want to look at things in different time frames, but we got to pick out something to start out with.

If we want to go to prestart, give me an option that ... allows me to change the time frame. (thinks we have one)

Q: How about a button that say's "prestart", "shut"

A: you can do that. define start as 0-6 (secs.); default to mainstage which will be defined five seconds from engine start to engine cutoff; and engine cutoff is always defined in the header; start phase is defined as 0 to 6 secs. from engine start; shutdown you can initially define as the engine cutoff command to cutoff + 10.

Q: ten seconds.

A: yes, some things we look at a smaller range, some things 20-30-40 seconds;

Prestart - lets go from -8000 to 0. That one is the one that varies the most. Not only do you have diff. chill times on the diff. tests, but you're looking at diff. things in diff. time frames but you have to choose something as a default.

Q: What I was thinking was up here we could have a buttons that could say, "PRESTART", "START", "SHUT", "MAINSTAGE". You have to have mainstage, ... and he'd click on that one first and then he'd come down here and click what he wanted he wanted, and however that was, we would default to how we plot.

A: I'd like to keep the option to enter in the manual time.

Q: We could keep the range as it is.

A: That's fine.

Q: This would be an example of the graphical reasoning explanation that Tim thought you might want. DO you want something like that?

A: I would come out with some words like this (current display) and then when I click on it, give me the option to pull something like this up (Tim's proposed display). The only thing is - I would also quantify the amounts. When you say Fuel side turbine temp is low - how much low? Because that could be a big thing. Low could be just a little bit in which case it doesn't matter much. If it's determined that it's low, it's determined that quantitatively so give me the number.

Q: I'm looking funny because it may have determined it quantitatively, it's (garbled) using that term qualitatively.

Erik: what does that mean.

June: that means that when the program is reasoning, it's just saying it's low, not by how much. and then later, it reasons whether that low, is a significant low, or insignificant low.

Erik: but if it's not a significant low, it shouldn't be printing out there, right.

June: No, I think it still prints it out as a significant symptom, it just doesn't see how it gets stopped. This is the leg that is correct, this one isn't.

(Tape: 684)

Maybe what we should do is let you see it like this first, then if you like something like this, we continue. If you don't then we don't have to.

Erik: Ok, but I still the idea of the logic path because it allows us to go down and say "oh this is how it got to the final answer." and that's going to be a big thing because we're going to go back and say, "what if I change this or change that". I've got to be able to backtrack somehow whether it's by words or diagrams, I've got to be able to backtrack to exactly what it reasoned to get the final answer. In the future what I want to be able to do is change hypotheses.

I want to be able to say, "What if the low pressure fuel pump speed wasn't low?" - I may have thought it was bad instrumentation or whatever, and have it recalculate the whole thing.

Q: you want to do "what-ifs?"

A: yeaahhhh. ... as a first step, all I want it to do is tell me here's the answer I got and here's how I got it. The step beyond that ... is what-ifs.

(tape: 720)

June: that would be a great mechanism for what-ifs. If you had this observation here, if you double clicked on it, you bring up the graphs. BUT if you single clicked on it would open up or pull down a menu for you that would have something like this graph (Tim's idea) so you could see it and if you wanted to change it then you could edit one of those boxes and ...

Erik: I like that.

June: Don't like it right off the bat.

Erik: I've got to have something like that because I've got to find out how it reasons to get it's answer. Because it may realize something I don't realize and I'll go, "Oh my God! I didn't even know that."

A3-173

... Calculations - good point. I want something to give me calculations.

Q: what do you mean by calculations.

A: $PID\ y + PID\ z / 17$ (etc) = plot it. I think pv-wave is going to do that

Q: you mean the plot package. pv-wave doesn't do that. We'll have to check with Jeff and Jean.

A: I just want the ability somewhere.

Q: No more druthers?

A: 3-d plots ... but don't want to get crazy.

Q: How about waterfall plots instead of multiscale plots.

A: NO, because on one plot, a lot of the changes we're looking at are from one data point to the next data point. ... I need them exactly lined up and the waterfall time shifts them.

If I could at all scale things with the pointer, I want to do that.

...
(tape: 783)

One other possibility. When it comes out with observations on the basis of certain pieces of data, is there a way to get the system, when it runs this thing, to give me hard copies of the data overnight as it runs.

In other words, it says, 'I've observed these things "

Q: I could imagine that as we go through the rules for that part of reasoning, that we could just start writing to a file that are impt. to that and then plot it out.

A: I like that because it gives me that backup capability that I've got hardcopy of the stuff that I could look at right away. I could page through that lot faster than I could bring up the system.

The other thing is if the system works through the night and is dead in the morning. ... power goes down ... system crashes ... I've got something.

(tape: 809)

...

Q: What kind of mechanism do you want built in so that it would automatically update the anomaly database? and how do you want to populate the database?

A: I want to be able to get to the anomaly database. In terms of populating the anomaly database, at least at first, don't let it update the anomaly database.

Q: So you want the anomaly, but you want only one person with the specific option to actually update it.

A: No. You could put the option on there I guess I'd have a separate screen. and on it is test A10695 and here's the anomaly and observations that were noted. And then have a button next to each one to add to anomaly database.

Q: The same way that we use ... in the anomaly database ... to build graphs ... click on it and it jumps to the other side but instead of graphs we're using anomalies and when you hit go, it adds to the database.

A: yes, that's fine.

Q: Only one person or very few people should have that option?

A: Probably. Right now, we have access to the anomaly database ... if you want to, put in an option for a password ... (have the) option to deactivate it if we want ...

(tape: 879)

Q: I was thinking that whoever logged in to start the session ... would have the capability to write to the database.

A: I wouldn't do that. I may log in but it may be Randy that determines what goes into the database.

Q: we could limit it to group ids.

A: That's fine. On that same screen, you have to give us the ability to put in anomalies that the system doesn't recognize.

Another thing is, ... when you call up new data ... through the info entered into the hardware database. I think that's done right now, but I'm not sure.

Q: and you want the ability to actually display what the data is. You only got 5 or 6 numbers, do you want them written on this (first ds screen)?

A: No. All I care about is where has that thing run before? ... What engine? - what test stand?

A3-175

KA Session with Erik Sander

Phases (4/29/92)

Tape: 000

Q: Help us out with controller phases, what you consider unique time slices - what is a phase;

A: one easy way to determine what phase the engine is in ... the Engine Status Word , that's PID 293 ... That's the easiest way. It has certain levels telling you you're in this phase or this phase etc. Have a program in C which does this already. We could give you a copy if you want it. ... Also, this program tells you when you drop LOX, or fuel, when you got presses. The best way to tell that is the LOX inlet press, the fuel inlet press. the LOX and fuel inlet temp.

If you want exact,... you want the press. When you drop LOX, it gives you good amount of head. (20 psi) ... drop the fuel and you get 1-2 psi. Not as clear as on the lox side.

Q: are the engine controller phases on this FID subjective or they always the same.

A: Certain levels are set, e.g. Eng. Stat. Wrd goes to 333, that's purge seq. number such and such, don't remember off the top of my head.

Q: when you're in Hydraulic lockup, is that an engine status?

A: I think your engine status changes for control modes in main stage, but I'm not sure - can find out for you. Another parameter that you may find useful is a FID counter in the controller that counts the # of pids that have been posted. If you ever need a pid number, ask Taylor. He knows them all.

Q: Failure IDS, does each one have a associated control action?

A: yes. and it ranges from no action taken to shutdown. A listing in the SW. spec.

Q: After Shutdown the controllers is down. Are there any phases associated with shutdown.

A: After shutdown the controller isn't really done. it's still determining when to do the purges.

Q: That's not Facility controlled then?

A: no.

Q: When in prestart, you said "so and so pushes the button" they decide in real time when to start the purges?

a: Yes, they will go ahead and ... decide when to go through purges 1,2,3 4. They give the signal to the controller to switch over to purge 1,2,3,4. When they're ready to start the engine, they will give the signal which is actually the engine enable. To do that, you have to be in "engine ready".

Engine ready is basically meeting all purge 4 conditions plus you also have extras such as LOX inlet Press, fuel inlet press - which all have to be within given ranges. When the controller senses that, it'll give signal saying Engine Ready. You can see that in the Engine Status Word.

Q: Do you ever double check the controller if something went wrong..

A: We will on occasion. ... Usually the controller always knows what it is doing. But you have the potential for programming

A3-177

errors.

Q: would you think it would be worthwhile ... to model what the controller needs to reach a particular FID. ... I don't think we would need it.

A: Typically, the controller declares a FID, it's real, or the instrumentation has gone out of range. ... In terms of programming errors, that comes more in terms of setting the sw constants C2, kf. by accident. ... it's been known to happen. (after conferring with Tracy.)

Q: Any qualitative differences other than faster sampling rate. or cleaner data between block 1 and 2.

A: Yes, there's differences in the way it controls and takes responses, whatever. The Controls and NASA people would have to give you info on that. In terms of engine systems viewpoint, that's all we see - not much else. You may get different reactions when you get into a failure scenario something like that. but nothing that I've found that major.

Q: You don't really keep in back of your mind the controller you're dealing with.

A: No, not really. In the back of my mind you only keep the controller as a Block I or Block II because of (1) the bit toggle (2) if we get into some crazy thing. ... e.g. On a recent test we had a hydraulic lockup and we failed the A channel on one of our preburner valves. It was toggling back and forth. The immediate response is to call the controller guys because they're the guys that know about this. They then get into, "Well, we had this Block I software and we now have this Block II configuration ..." and we get into these kinds of things.

Q: We have to get into the things that can't be identified by the PIDS which sounds like they would all be in Prestart. Is there something from Prestart where you can get these times. How can you find, when you're looking at the data, when you're in a certain phase?

A: Again, the most obvious thing is the Engine Status Word. because it will always tell you what Purge Sequence you're in and it will tell you're in Engine Ready and it will tell you when you hit Start Enable.

Q: Is there a significant lag when they see conditions are OK to start purge two.

A: A lot of times, they won't get data for Purge one or two. Purge one and two are five minutes apiece. They're very small. (Showing SSME pocket data, pg 1-11.) Tells here is a ground nitrogen this is what they call the vehicle helium. On development tests on the ground that comes from the stand. Grnd. Nitrogen comes from the ground for flight and the helium comes from helium bottles on the vehicle and this is one of the other....

(TAPE: 100)

In addition there is a package here. ... (summary of eng. purges, lcc and flight.) ... Description of where, why, what to do, where to look. etc. Typical events on launch ...

Purges 1&2 are typically small 5-10 mins. Purge 3 can be hours. When we're in flight, on the pad, purge 3 is the vast maj. of time. 8-10-12 hours. Just sitting there - that's when the engine is really content. Cryogenics are down and we're purging as we should be and then 4 mins before flight we go into purge 4. We can hold in purge 4 for 12 mins. but we typically launch at 4 mins. ...

This (summary book) also has schematics of PCA Pneumatic Control Assembly ...

The PCA contains the valves that allow the shuttle bay to work them

and provide purges to different parts of the engine. (referencing ssme orientation handbook) That's the piece that translates from the electrical controller signals to allow the signals to go to other parts of the engine. It's Pretty complicated.

Q: Other than control modes, are there any other things that you think are significant when you're analyzing data.

A: Prestart, we're looking for a couple of things. First, do any of the valves leak? You tell that by the skin temps and the internal temps of the engine - the turbine temps, MFV skin temp, the OPOV skin temp. Those tell us if those valves are leaking. You'll see those temps drop drastically.

Additionally, you're looking to make sure that the engine is being purged properly. That means that (they run) according to that schedule in the (pocket data book.)

During purge 3, which is the majority of the time, you've got several things going on. You've always got the intermediate seal being purged and you have ??? being purged with nitrogen. You've got two preburners being purged with nitrogen and you've got the three minute purge every hour just downstream of the Main fuel Valve. ... and you got pressure transducers to show us that we've got those things going on. In addition you've got the valves being held shut by pneumatics as well as hydraulics. When we go into purge four, we pull the pneumatics off the valves. In fact there's an LCC reqt. that the valves shift by .1% because up to that point, we have the valves being rammed into the seat and now they're being released just a bit. ...

The HPOP intermediate seal switches over from nitrogen to helium. Remember that yesterday, when all pressures related to helium shifted down a bit because now we're feeding the intermediate seal as well, any thing associated with nitrogen pressure you saw raise just a little. Basically that was just the two preburners.

Then when you get into purge 4, and on the ground, it's not at a specified time, you cut the nitrogen to the preburners. (referencing the Countdown Key Vent Schedule.) this thing is very rigidly set on launch. Basically anything from t-9 mins to launch is very rigidly set because it's controlled by the ground controller. That's not the case on development testing. That's why we don't compare prestart info to another test. Prestart varies so widely and you can't plan on it ... (facility people) work this problem, work that problem.... They're going to keep working things until they think it's safe and ready to run.

On flight everything is rigidly controlled so (e.g.) nitrogen is turned off at a given time. The launch is much more repeatable and is much more comparable between launches. But in terms of ground the only thing you fix in time is when you (they in facility) send the signal to the engine start. What they're really doing is sending the Start Enable. 4 seconds later, assuming the two bleed valves close and the POGO RIV opens, we'll get the engine start signal and the engine will start.

Q: Are there other identifiable phases, perhaps even in mainstage, - e.g, if something goes into pump cavitation, would you look at that as "oh, I'm cavitating now" and keep that info when you look through the rest.

A: We wouldn't consider that a phase. The only phase in mainstage are electrical or hydraulic lockup.

If we can still control the valves but have some kind of a problem, we'll declare a phase called an "electrical lockup". What we're doing then is locking up the valves at their current position. We're doing it electrically. In other words, the controller is still sending signals to the actuators to tell them to hold the valves at this given position.

If we lose hydraulics (if we lose an AP or something like that)

A3-179

where we don't have control of the valve because we don't have hydraulics, we go into a phase 4 type "hydraulic lockup" All that does is within the actuator we have a couple of valves and one of them will shuttle over and literally lock the hydraulics into the open and closing piston on the valve itself so ... then the valve shouldn't move. It's important to note that in hydraulic lockup, the valve does move - it drifts because of leakage out the open pistons sides. The valve will slowly drift down.

Q: In both cases, the engine is no longer in closed loop control?

A: that's right. You've got basically a fixed orifice engine. and you should have that fixed orifice at a given point, but you don't because of the clearances you have on the valves ... allow some hydraulics to leak out and the valve to close slowly.

(TAPE: 202)

Q: You mentioned that you want to make sure the pumps are primed at a certain time from each other. Is that controller controlled?

A: The only handles the controller has on the engine at that time are the valve positions. We set the controller valve schedule to try to prime us at the right times and the right sequence with the right deltas between the primes. That's how we control the prime times.

Q: Does that change test to test?

A: Sometimes it does. Especially some parameters, one parameter called the OPOV open loop command. If you look at the sw spec, there are different positions that the valve goes to. One of those, we pretty often change. We move the OPOV command at a given time up or down to try to speed up or slow down the LOX side a bit. It has an effect on the prime times.

Q: Can you tell if something is primed by a certain parameter.

A: Yes.

Q: If you did and noticed that they were shorter... than some time that you want, you would notice that?

A: Right. The spec has a prime time rqt. for each preburner. We've got those documented as well... It says that the fuel preburner will prime ... at this time, the main chamber so much after that and then the LOX preburner at this given time.

Q: Is the only thing that causes them to be primed the valve position or if they're at a higher temp. or pressure?

A: No. Which pump you have on there; which engine; what are the resistances in the engine. All of that.

Q: If the schedule went according to what you had set, would there be anything happening in the engine to cause you to prime early?

A: Yes. You could have hot or cold LOX; hot or cold fuel. The prime time is a fcn. of many different things: the LOX; engine resistances; turbopump performance; all that kind of stuff. The software is the best handle we have on controlling the prime times.

Q: If we were to try to break down and code when the engine had primed, the best thing to be would to look at a certain combination of parameters.

A: We look at the parameters specified in the sw spec. It's arguable that these are the best parameters. But they are the most repeatable in terms of we got the history on those parameters. Now the fuel preburner is the first to prime and we look at a given time slice and when we see the rate of change of fuel preburner change by a certain amount, ... (we have all this in code)... we declare the fuel preburner primed because that's when we're getting real energy to the fuel turbine and the pump really begins to pick

up speed.

The main chamber is declared primed when the main chamber PC crosses 100 psi. When we saw it yesterday it crawled up and then it all of a sudden it will go "clunk" and it spikes up there.

The LOX preburner is declared primed when the LOX turbine temps,... either of them ... first occurrence, go from a decreasing rate to an increasing rate. That means you've got fire basically.

That sets by definition when you've primed the different chambers. Again it may not be the best indicator. There's questions out there like, "instead of the LOX turbine temps do you want to use the LOX turbine discharge pressure coming up quickly?", something like that. But it's at least consistent with what we got in the past and what we've got in the spec.

Q: Whenever either of these things have occurred, the chamber would have had to been primed?

A: Pretty much.

Q: And the same, it couldn't have primed if this didn't happen.

A: Right. They're real good indicators of when it's primed and then again we have all that in software already built in.

Q: Are there any other phases that you're concerned about?

A: On shutdown, there are two different phases. We normally hydraulically shutdown the engine. What that means is that simply we control the valve through hydraulics. What we'll do is send the valve to given positions during the first 4-5-6 seconds of shutdown and what we're trying to do is shutdown fuel rich. The last two valve to close are the MFV and FPOV which run the fuel pump and allowing fuel to go through the engine.

When we feel we don't have control of the hydraulics, or we can't control the valves hydraulically, one or the other, we shut down in the mode called pneumatic shutdown. In that, what we do is ...let's look at a schematic ...

Q: Are they mutually exclusive?

A: Yes, you have either one or the other. ... Here are the valves ... here are the actuators. ... There's the main fuel valve actuator ... it's a good sized piece ... The actuator itself has an open piston side and a closed piston side. ... Normally there's a piston contained in this cylinder. It's got a little arm, and if you want, think of the valve as being right below them ... You've also got a little ram back here which is really another little piston that's in here. It's what we call the pneumatic shutdown

(TAPE: 281)

piston. What happens is that when you go into pneumatic shutdown, you assume you don't have control of hydraulics anymore. SO you actually allow hydraulics to escape from this side and you apply pneumatics to ram that thing closed. what that does to you is that where you normally have a shutdown, the valves are modulating back and forth. Now you have one where you have the valves coming down in a straight line fashion. It gives a much different trace on shutdown. It's still a safe shutdown, but it's a much different shutdown than what you get hydraulically. ...

Q: Could you also detect this through a FID ... PID 293?

A: Again, I don't know that PID 293 will tell you if you're in lockup. I think it will, but I'm not sure.

Q: Is the FID different for each of the phases like Prestart, Start shutdown ...?

A3-181

A: It's different for them, but for mainstage itself, between start and shutdown, I'm not sure if it will change when you go into hydraulic lock up. There's other ways to find out. You declare a FID that's probably the primary way to know.

Be aware that you can actually pull hydraulics off the engine, and the ... controller has no control on the valve but it won't know it. It won't declare hydraulic lockup. We normally do that because what happens is we pull the hydraulics off the valve is just sitting there. When you pull the hydraulics off the engine, ... what I mean is you drop the hydraulic supply pressure. You just kill the hydraulics from the facility. The valve will physically lock up but the controller doesn't know it. And the reason is that the valve hasn't drifted 6% off from it's commanded position yet. So we'll spend a good amount of time, 30-40-50 seconds, depending on the drift rate, at a given position when we're actually in hydraulic lockup and the controller doesn't know it and it hasn't declared it.

Q: And you can tell that from a valve position being stuck and then all of a sudden acting normally?

A: No. If you pull the hydraulics off, up to the point that you pull the hydraulics off, the valves will be controlling normally. Controlling normally means controlling to power level transients and inlet conditions and that type of thing. When you pull the hydraulics off, the valve will essentially just sit there. It will start to drift a little. When you declare hydraulic lockup, it could have either drifted off 6% in which case it would go to the "B" channel. ... It still hasn't declared hydraulic lockup. You have to get 10% error on the "B" channel.

(TAPE: 318)

Q: I'm wondering if this is significant because when you're looking over the data, you know when you're in hydraulic lockup.

A: You pretty much know it. One of the first reasons we know it is that we usually schedule pulling hydraulics off the engine. We're doing the test and check that so we know when hydraulics are off the engine. Another way to know is to see the valves start to drift and stop controlling.

Q: If we put into a computer that if a valve starts drifting for a while and ... I'm assuming that once a valve goes into pneumatic, it starts controlling

A: No, ... once you pull hydraulics off the engine, you will not get normal control again. The only time when you switch to pneumatics is when you physically shut it down. That's the only thing the pneumatics can do. That's the only thing the pneumatics can do is shut the valves down.

Q: And they schedule this sometimes and it drifts and then what?

A: It depends. Usually it sits there and drifts away and you just run the engine and the engine will start drifting in performance - it'll run to a different Mixture Ratio - that type of thing. What we've seen lately, especially with the high leaking valves is, that is that the LOX side will invariably power up and you may hit the LOX turbine temp limit, which is what we did on the last type of lockup test. The lox side just powered up and up because you weren't controlling it anymore and you just ran it to the lox turbine temp limit and then you shutdown and then you pneumatically shutdown because by then you had declared hydraulic lockup. But the pneumatics cannot control the valves on mainstage. The only things the pneumatics can do is shut the engine down.

As an example. Here's the normal valve schedule for shutdown during hydraulic lockup. We bring the MVF out; we start closing the MOV we start closing the OPOV. The CCV is way out here, the MFV is way out here. We don't close the MFV or CCV until after 5 secs. Where with the lox side valves, the OPOV and MOV were closed before 3 secs.

In pneumatic shutdown, you see these all as a straight line. All you're really doing is ramming it on one side and releasing the hydraulics on the other side. You don't have any variable control to hold the valve in one position or anything like that.

Q: You pull the hydraulics down ... the controller still thinks it has hydraulics and is still trying to control, so we might have some indication from the controller that it will still be trying to manipulate the valve.

A: That's right. And what happens is if the valve starts to drift closed it will send the command to open. For example on the FPOV, let's say the PC starts to drift down. It'll say, "Hey I don't have PC" and electrically it will continue to try to open the valves. But electrically all it is doing is sending a little wand back and forth to the actuator. It doesn't have hydraulics to work with. So it's doing it's best to open the valves but nothing is happening. Pretty soon you have a 6% difference and you switch over to "B" channel. When you switch over to B channel, and this is important, you don't start at the 6% miscompare. What it'll actually do is reset the command to the point where B is at right now and then when you get 10% from that it says, "I'm gone."

Q: ... Both controllers?

A: Yes. In terms of post shutdown phases, when you're doing pneumatic shutdown, you'll affect the way the purges are put on the preburners - that sort of thing. So you can tell that.

If you really want to know when you're in hydraulic lockup, there's "real life" hydraulic lockup and "controller knows it" hydraulics lockup. As I said, you can pull the hydraulics off the engine and for practical purposes, you are in hydraulic lockup.

Q: The only way the controller knows that is, if it commanded it.

A: No, if it failed the valves. Once the valves drift 6%, and then 10%, then it says it's in hydraulic lockup. In reality, it's already been there for some time. In terms of the engine operational changes nothing happens, because you've already lost hydraulics and you've taken its "arm" away, it's ability to communicate to the engine. By pulling the hydraulics off of the valves, the controller has no control over the valves anymore. We do that back here at time "A".

Q: They're pulled off all valves?

A: Yes. You can't pull them off just one valve. You pull the hydraulics, you pull all the hydraulics off; the valves go into lockup, that's down here at time A. At that point, you've taken away the controllers ability to modulate the valves and control the way the engine operates. The controller still thinks it has control, but you taken away it's arm to the engine. The thing is still sitting there electrically - it's still controlling away, but nothing's happening. The valve's start to drift. The controller starts to see 2%, 3%, 4%, 5% and tops at 6% and switches over to B. Nothing happens in the engine because again it has no control. Then it goes 10% on B and a flag goes up - hydraulic lockup. The only way that affects the engine operation is if it does not declare a hydraulic lockup before engine shutdown command, it will try to shutdown hydraulically. That's very important because what it will do is give a very slight delay in your shutdown and if you're trying to do a shutdown comparison vs. other tests you'll see this slight delay because what it'll do is command the valves closed and it will take about .2 secs for it to realize that the valves aren't following the command. It could take more than that depending on how far the valves have drifted.

Let's say the valve is 9% below the command, then you hit the shutdown command. Then the command has time to go 9% plus another 10% on the other side ... before it realizes it has a problem and declares a hydraulic lockup. At that point it'll ram the pneumatics into the closing piston and shutdown pneumatically.

A3-183

Whereas if it has already declared hydraulic lockup in mainstage, it'll immediately know when you declare the shutdown command it doesn't have any hydraulics, therefore shutdown pneumatically.

Q: In shutdown is there anything significant? - or in post-shutdown?

A: In terms of phases. The only thing is whether you shutdown hydraulically or pneumatically.

Q: You don't care what goes on with the post-test purging or ...

A: No. Post-test drawing and stuff like that, we don't even get the data. We typically only get data 300-400 seconds after the test after the shutdown command.

(TAPE: 422) (end of interview)

Interview Session Date: 5/1/92
Brian Piekarsky

BP - You set the fuel flow and then the lox flow. In reality the controller would be [set up to] whatever MCC chamber pressure [is at], and so there is a correct fuel flow for your particular hardware such that your lox flow to MCC PC would give the right mixture ratio. What we will do [is], we'll look at the mixture ratio and whatever is up from 6.011 is a C2 correction that will change the fuel flow. The right fuel flow so whatever lox flow is required MCC PC with that fuel flow will be the right mixture ratio. Models of all that stuff will use a form of these equations to figure out the C2 details. You can do it by hand along with all the other stuff.

Question: Yesterday, somebody showed me approximations Rocketdyne had come up with to approximate hydrogen property tables of certain pressure and temperature ranges. Do you use those or actually use huge [property tables].

BP - There are property tables in the model. The bad thing about that is they probably go back to 1970 or earlier. They haven't really been updated, but there is property data in the model.

Question: We have a version of tip 88A. Is there a more recent version of the model?

BP - They came out PBM90A. Then they came out with PBM91A. We are currently implementing PBM91A. [They are] suppose to come out with a 92 model, but in reality they won't make it until 93, but it will still be called 92. It will probably come in the spring next year. The 90 [model] was a good one to have. It had enough changes, and we [have] been using that one for quite a while. The 91 [model] has some specific stuff on how they tag engines. We sort of came out with some new procedures for tagging engines over long time slots. They made some changes to the model to facilitate that.

What we do for a flight engine is; they will run an engine, run a standard acceptance test, and we take that data and reduce it over a particular time. In the past its only been [a] 10 second time [section from which the engine would be tagged]. The particular time in the test [would be selected] and [we would] get all its performance and stuff [from that time section]. From that information we [would] make a prediction out of standard conditions and that is it's [the engine's] tag values. We do that with all the flight engines and we predict them at the standard conditions so now we can compare "apples and apples". So if a particular acceptance test's ran way off mixture ratio we can still make a prediction [about] what it would do at a nominal mixture ratio and we compare all these flight engine performances at that condition.

You may hear [the] term, rating. We rate [an] engine to standard condition so we [can] see where it lines up with all the other flight engines. Now we always had heartburn with that 10 second slice that they chose. It took a while, but we found some reasons why that particular small 10 second slice can be affected by certain things in a non-repeatable way. So what we asked for was a longer time slice to get your [engine] performance from. Mostly there are two different kinds of acceptance tests still run, I think one of 165 [time] slices [where] we take data at 165 seconds and average that to get one point. That averages out the affect we were seeing previously.

There were some other real intricacies that require some changes. Some of it was stuff like since your averaging so much data you needed double precision variable versus single precision, and we started averaging some redundant PIDs rather than counting on one. Even though there are real subtle differences between two measurements, there is really no logic to say this one is any better than that one. They had come up with a spec change for the new way of tagging it, therefore they had to change them all. That was 91A [model].

A3-185

[Model] 92A should have some changes based on TTB data. What they typically do is take the power model's generic prediction code and they make it predict the average of all your flight engines at nominal performance at nominal conditions. So your nominal parameters give you that average temps and since they released the model there are seven or eight more flight engines to put in the database. So that [is] one reason they want to release a new model. And the other one [reason] is the information from the TTB, some of the internal stuff of the engines are not what we thought they were. [And] we never had certain information, [so] we were bound to be off somewhat on somethings. We are trying to take as much as we can learn from [TTB] and stick that in the model too. That is what they are shooting for in model 92.

TIP88A isn't a bad version of the model though, if you go back a couple years before that then your in trouble. I wouldn't recommend you use that anymore, but if you're using TIP88A you're not far off.

Question: Do you only tag and rate acceptance tests? Or do you also tag all the other engines too?

BP - It depends, sometimes we get requests to tag other engines. Especially these funny configurations that [they have] been coming out with lately; baffles, phase II+ and large throat. They want to know how its performance compares to the typical flight engines, so we'll rate those.

We're getting ready [to] recreate all our databases with 91A. One thing that we decided to do is to create a sort of ready database for more of the tests. So what we actually do when we have a new version [of the model] in, we have an automated way [of] rerunning all the tests with the new model and recreating databases. So with time what we're going to do, in addition to what we normally do, is rate a number of tests as well, other than [just] acceptance tests. Any tests that had significant 104% powerlevel mainstage, [that] we can run [the] extended time slice [calculations for], we can come up with some rated performance and have a database for that [engine].

Question: One of the things that maybe a problem is trying to figure out if you have a [efficiency] change between [a] component in one test and [the component] in the current test that your working on, is it do to venting conditions or is something else going on in the test that [causes the] change in efficiency. Do you feel real comfortable with delta [values] you would get from running the model with the different efficiency change?

BP - Unfortunately, the answer I have to give you is that I've been here for five years and it has taken that long to know how to answer that question. It depends on the parameter, and that's what has taken five years. There are some things I trust in the model and somethings I don't. I don't trust valve position analyses in the model. I trust pump efficiency, especially in the fuel side. I don't trust turbine efficiency. A lot of this is not the model's fault, we don't have a lot of instrumentation on the turbine end. So you don't know what the inlet temps are, the inlet flow rates or the discharge pressure, and even the turbine discharge temps you have coolant mixed in there. It not very repeatable so its hard to get an accurate assessment of what's really going on in there.

One of the things we are often asked to do when we see shifts and changes and stuff, is [to] do our best to explain [them]. Somethings [we] are more comfortable with [than others]. We express [our explanations] in confident levels, [so] we can suggest this change is do to this, but [we are] not real confident that it could just as well could be two or three other things. There are other times [where we've been] pretty sure that something is caused by certain things. In the last two years we have learned a lot about two effects, venting, lox venting in particular, and propellant transfer, [and their] affects [on] engine performance. We have always known it [venting] affect the pumps, but its surprising how much it affects the whole system. It depends on what it is [occurring], sometimes we can give a much more confident

answer than other times.

Question: Are you uncomfortable with the absolute value in efficiency that you get, [as] opposed to [an] incremental change? [In other words], do you feel more comfortable with not so much the absolute value but the [incremental changes] that the model gives you?

BP - A lot of the things we are asked to do is not speak in terms of absolutes so much. Some of the measured parameters, like when we have to do predictions for turbine temps and stuff, we'll have to give our best shot at what the temperature is going to be, it's absolute temperature. But allot of the time they'll see a temperature change in a test, say turbine temps changed 40 degrees, and they will want to know what brought about the 40 degree change. So we play with things in the model and try to give us [that] gain. We are more confident in gains with the model than we are with absolutes in a number of cases. TTB gave us some information that should help us a little bit with some gains [in order] to know how far the power balance [model] is off on some of these things. We are often asked to look at things in terms of deltas rather than in absolute [values]. And people pretty much know that around here. Something that we are often asked to do is, what is the gain versus going to max to min lox pressure on discharge pressure. Pretty confident in those. The answer we can give is a reasonable one [and] we're not going to be off or in the wrong direction.

Some people would probably ask the question too, what is the gain of switching it, having a new [component]. They wouldn't get into the LRU number or anything.

Question: You would have a gain in general in switching a pump. What would be the maximum [gain value] you could accept?

BP - It depends, that's my answer to everything. What we can do sometimes is take the last [test firing], and we have to do this for flight all the time. They will test a pump over on another engine and bring it over here to this engine and want to know what it does. The same way we tag engines, [we] will reduce the test [data], get some hardware characteristics and use that to make prediction. What we'll do is reduce the test over here and get some characteristics as much as we can from that pump, put that into a prediction with new hardware. So if we know a pump is pretty much a dog, we can use the information and predict how its going to do with another unit.

The fuel and lox turbopumps offset each other somewhat and you'll have a real hot burning fuel turbine [that] can drive the lox turbine temps down. The model has a way [of] using the hardware characteristics to sort of match this thing. Because the turbine efficiency and the turbine ends stuff is so hard to really deal with, [since] we don't have allot of measurements, that usually accounts for our [low] confidence level. What we are trying to do is we have different confidence limits. Typically flight predictions, even though we grab one pump over here grab another pump other there and bring them together for the first time, we still tend to predict within 50-60 degrees on turbine temps. People sort of know that by now, we [have] been doing it for 4-5 years. Every now or then one will be off more than that, but typically they are within that [range], and sometimes they are even real close.

If the hardware was all tested together with the engine, prediction tend to be really good. Sometimes that's not saying allot though, people tend to look at how the test performed and they have gains in their head. They can do a pretty good job to guessing how its going to do anyway. The only time it really helps you is if you are going to change operating positions quite a bit. If you were off the mixture ratio, it [will help change the value of] C2 to get you back on mixture ratio. That kind of thing.

Question: I think I have a very thin user manual for TIP88. Do you have a procedure of how you set up the data reduction routine for a typical test? In what your looking for? parameters?

A3-187

BP - Not what we're looking for, but what we have is a standard checklist of the things we do. I'll get you a copy of that. What we may want to do is probably try to explain a few things before, the checklist might be a little too personalized for us. We may know some of the slang words, I haven't seen in a while. But it is all the steps we go through from finding out a test ran to what we presented the data review. Someone not too long ago did a TQM, where we flow charted all stuff we were supposed to do. That is Martin Marietta's activities, what we do for data reviews. What we did is tried to flow chart or at least put together a step by step type of thing of everything the data guys do [and the] model guys do to support a data review. We tried to put it in order, I haven't seen the final form of it, and I'm not sure exactly how it came out. Last I saw of it, it had even detailed steps, like it would pick a pretest package [and] give a copy to the model people, but it doesn't tend to over look anything. It will tell you everything we know.

Question: Do you think it would be worth while to get a new version of the TIP model? We may have some people in our group that might work with gains.

BP - What are you running on?

Response: Right now running on a VAX. We have some unix systems too.

BP - We go through some headaches because Rocketdyne runs theirs on Perkin Elmer [computer], so we have to do all kinds of crazy things to get working on the IBM, and if you had an IBM we could probably arrange to give you our latest and greatest, without much of a headache. Well, allot of things you have to do, when we first compiled the model we get from Rocketdyne, we get anywhere from 1,000-2,000 errors. Some of them are things you can fix and it will eliminate a couple hundred errors at a time. They [Rocketdyne] have a compiler that's is very generous. They have syntax errors I've been trying to get out of the program for a while, and their compiler will let them go [through]. If you got it from John and it originated from the IBM then probably all those things were fixed. We had to put in our own I/O routines.

I wouldn't think there would be any problem getting the newest model. As a matter of fact, we could probably get you PBM91A. It would be worthwhile to have, but you would have to go through that headache again. We had to do [that] every time a new model comes out. If you talk to John, you could probably get a copy of PBM91A and that's [what] we are about to start using for everything. That's about as current as it gets.

Question: Did they come out with any new user manuals?

BP - There is one for PBM90A. There's a users guide thing that has a list of all the variables, there is about 1,350 of them. And for some reason they didn't update that for 91A, but they did put out user requirement document that shows some of the description of the changes they made, and from that you can figure out the changes they made to the A array [This is an array, a rather large array, in the PBM], which you need to know about. It let you know how the users guide changed. What you can do is take the copy of PBM90 users guide and a copy of PBM91 requirements documents and thumb back and forth between the two, that's what we have to do.

Question: The only other question I have maybe to deeply involved [for this interview], because I did do a little [work] with TIP88. There are some parameters in the input deck, I think they are called multipliers. Is there a nominal value for each one of those multipliers? Or is 1.0 the nominal value, because some of them came in at .97 and some came in at 1.02.

BP - Probably a long time ago, 1.0 was the nominal value, but what they have done is as they have rebased the generic power balance. They baseline it, give the average value of the flight engines.

So [for example] turbine temps, lets say for instance average turbine temps 50 degrees are higher than what you make [simulate with] the power balance. What they do is change the nominal multipliers to get you that 50 degrees [difference]. they play with all these multipliers. They'll play with some resistances, [and] some other things in the model, to match the data and baseline the data to it, to a particular data set. So that's why they [the multipliers] are not 1.0 anymore. You get these weird numbers like 1.0287 and (end of tape)

A3-189

{Tape 4, SV1 Side B}

{Marc Neely on Sensor Validation}

Side B: Sensor validation

{June explaining to Marc that this session is for sensor validation}

Q: ...If you can maybe cover the philosophy on sensor validation and how you go about doing it, would be helpful.

A: I don't think you would try to validate all the sensors.

Q: Up front you mean?

A: Right, unless we note something is significant. I'm not sure if that is the right way to do it. The model group validates all their sensors before they run the model.

Q: Do you know how they do it?

A: I think that they apply a zero shift program to it, but I'm not sure about that either.

Q: Who would know about that?

A: Brian Piekarski,, or Bill Green or maybe Tracy. There some sense of validation.

Q: They probably only have to validate all the PIDs that they need to go through their model?

A: Right.

Q: So, it's probably a subset of the total sensor suite?

A: Right. There is a little different approach on (TTB), I think they try to validate sensors. A lot of them are unique, one of a kind instrumentation so they have a different objective there. Our objective is making sure there is nothing in the engine that would keep it from running again; that there is no bad condition or damage caused by that test, or something that could fail in subsequent tests. That's our primary objective. Looking at the data here, this is more data acquisition then research/ technology.

Q: They are trying to measure things that have never been measured before...

A: Exactly. When we do have an anomaly that we are trying to work and it's not obvious, then I guess you could always ask the question; "Is it the instrumentation or is it the engine, or real physical phenomenon?" We have several options, depending on which parameter we are looking at. Where we have redundant channels we can just clear the two data screens. By redundant channels I mean that we have a lot of instrumentation that have one sensor (one transducer) with a dual bridge. That's an easy way to do it. But you have to have a feel for what is an acceptable delta between bridges...

Q: Is that (the acceptable deltas) tabulated somewhere? or do they just know it?

A: There is not much delta, the only time I got concerned with that was when I was looking at low pressure fuel pump discharge temp. With this PID, an average delta is about .2 degrees that an acceptable delta of around .5, these are just examples, and the only reason I cared about that is it is used in a density calculation to control engine flows. But if we had a delta of 10, 20 psi on two channels of a 3400 psi transducer, ...?

Q: Now that you have two fuel pump discharge temps and lets say

A3-191

an, for just this example, acceptable delta is .5 and you observed .6 how would you know which one was wrong?

A: I don't know, unless you could back it out, if you recalculated the density and the flow rate, I don't know. On a flight engine you couldn't, you wouldn't have any independent data.

Q: From the facility...

A: For the facility, you may have some independent data that you could use to calculate a mass flow rate, but you have to rely on previous test experience, because even then you wouldn't know if it was the facility or the engine. Again, I have to stress that there is a threshold where it doesn't matter.

Q: In other words, it's almost normal?

A: Yes.

I was talking about anomalies, when we see an anomalous condition and we don't have the two bridges, one of the most basic ways is related to physical parameters, (in other words, if I have see a rise in low pressure fuel pump discharge pressure) I immediately look at the speed. If I see a rise in the speed and the discharge pressure, I feel comfortable that it is real. They are independent parameters, from an instrumentation standpoint, and they're related from a physical stand point. Another thing that we don't do a very good job of is when we see an anomaly that we can't verify with another parameter (we have some of those) is go back and look at pretest and post test to see if there is anomalous behavior in the parameters. It's a lot easier to tell when something is acting up when nothing is going on. And that's very easy to do.

The types of problems you can have is with sensors that are not calibrated properly. In other words, instead of reading 7,000 psi it's reading 4,000, really you just validate that by looking at related parameters. The more subtle calibration errors, I don't know how you get those errors.

Q: When you check redundant channels does that include CADs and facility comparisons? Is the facility measurement just another bridge off the same sensor?

A: Yes, but depending on the anomaly. If the anomaly takes place over milliseconds or 20-40 milliseconds you get into a timing problem. The facility data is not recorded with the same timing system as the engine controller data. That is not really a factor, our major incidences all take place over 10 milliseconds, that becomes a problem (I don't think this is what he wanted to say). That could be a next step (in our development of the sensor val. module), first look at general engine performance and then major incidences where you've obviously got a problem. The time issue is a big deal, when you get into a real small timeframe and comparing facility and CADs data. In fact, comparing the CADs to CADs data on real small timeframes is a little bit of a problem, because they are not all measured at the same time, but they are posted with the same time stamp.

Are you aware of stale data?

Q: Could you tell us which ones at stale?

A: Yes, I've got a list. Generally, when you have a stale and non-stale parameter the one with a "1" on it is stale. If you have a 40 and a 140, the 140 is stale.

Q: But not always?

A: It's not always, because p2, (average of two bridges in the transducer, then the averages are averaged), two of those, either A1, A2, B1 or B2, is stale. If you need a stale table, I think I've got a stale PID table.

{Claudia asking questions about this}

It's the same data, the controller just holds "it" for 20 milliseconds, ...{can't hear}

There are some hard limits for sensor validation, the controller has what amounts to sensor validation for all the redline parameters and launch commit criteria. They're automated, it doesn't let any parameter exceed a reasonable max or min value prior to the test. You would have to reconstruct the controller qualification limits in your model.

Q: Wouldn't that come out of the fids, we would probably have to translate those codes?

A: Yes, it would come out of the fids and if you're watching the fids, there are a couple of fids that give you failures, engine status word, etc., that you could pick up, it gives an integer representation of a octal, which you would have to translate into what the fid is on, it may be just as easy to reproduce the controller limits.

Q: When it comes back on the quick look sheet, it lists the fid on the top, does it just list the number?

{Agreeing type sound from Marc}

good its just the number.

A: Well unless the guy that recorded it looks it up.

Q: Those numbers are in the controller document.

A: There is a better copy of that. This may be an old Block I copy.

{I have that}

The ED people have a 40 page break out of all the sensor qualification criteria and redline criteria.

Q: What is ED?

A: They are the controller software people and the controller hardware people, instrumentation is over in ED.

Sensor validation, the last thing I can think of is that you'd need it to have an experience database. We have some sensors that have characteristics that do not operate correctly; they've done it forever and we understand why they do it, we just disregard it. For example, fuel preburner chamber pressure used to steadily drift down when we knew the pressure was constant, because of thermal effects. Some engines have an insulator block, a remote mount on the sensor so it doesn't drift.

Another example would be main chamber...

Q: Hot gas injection pressure, PID 24?

A: No, it's real.

Q: It read 14.7 a lot of the time.

A: That's when it's not hooked up. You could look at sensor validation, one part of it, is as knowing what instrumentation is there. {Can't hear}

Q: That would be listed somewhere in the pretest document that that instrumentation wasn't going to be hooked up?

A: Yeah, a lot of the times you have to go back to the first test of the engine, because that's where they give you an engine configuration.

A3-193

Q: What's the component that tracks the engine? A component that stays with the engine, doesn't get switched out?

A: There's really not one. In general, the powerhead, but that's not always true.

{can't hear}

On the lox preburner we had an icing problem, where the data always looks like this then it goes into a straight line, during a shutdown it may go like this and may come back in..now that's one that has happened over and over in the past. I don't know if it does it now. Now, we've also had other parameters where we've seen this, and we rationalized that it was the same sort of phenomena that we saw in this parameter. In short, there may be a way to database sensor failure problems for future use that can be referred back to.

Q: So I care about a sensor going flat at about the same magnitude, it could also go flat if there were hard failures so that we could assume...

A: In those hard rules in the qualification criteria, what they do is set the criteria at (maybe) the upper range of the sensor. The turbine discharge, the redline parameters, they have mostly an upper limit (but the lox side also has a low one) and the way they are set up they have qualification criteria that bracket the redline, so that if you come up in here and dwell in this point and this band or this band or three major cycles it will shut down, if you pass through that band and come up here then you disqualify the sensor. I can't figure out how a this could be real and "it" still be in one piece. So that's how that qualification criteria works. There are other, and its in the software also, the bit toggle on the sensor, which gets back into the data characteristics, is there the range is there, and if you see one do this {scribbling}, than it is out of range or that it is topped out. That happens, in fact, it happens with the fuel bleed valve position on almost every pretest, when you chill, the valve is open to 100% and you start loading propellants and the temps dropped and the valve position creeps up and then it changes data characteristics and its reading, and its bit toggling like this, across a flat line. Its at 106%, its just topped out, its full open and the thermal effect causes an electronics to read the top of the range to the sensors. But it doesn't have an upper qualification limit.

Q: Is there a select group of PIDs that you normally validate and others that you don't? For instance, when they get through a data book in the instrumentation package and that seems to concentrate on these redundant channels A and B, and also CADs vs. Facility. Do we just look at the PIDs in the package and say that if we validate these, these are sufficient?

A: I think that the logic behind that is when I was setting ? was to validate to redline and launch commit and the engine ready, those types of parameters that we don't look at once we start the engine, but we that have to rely on to start the engine for the next test. In other words, there are a lot of parameters that are not used during mainstage that are used just in prestart to check through leaks, purges, those type of things, and if they were to fail completely during mainstage, you wouldn't know it,

Q: Because you wouldn't care? You don't look at them during mainstage?

A: You wouldn't care as far as that test goes but for the next test, you would. Or, you would get into an engine test where you would probably do an instrumentation checkout before you even try to test again, but just knowing that there was an instrumentation problem would give "them" a big lead time. The parameters that have a direct influence on the engine I guess (the ones that we have the requirements on them) are the ones that we look at and

make sure that they are still okay. A good example is the main fuel valve skin temps. We used to have problems with it but we don't anymore, but it is just bonded to the outside of the duct and wrapped and taped. We used to detect leaks and it used to be bonded flight a lot. When it didn't bond, it just read ambient, we lost it. It doesn't have any affect on flight, once it is started you don't care...the main fuel valve is open.

Q: Before, we started talking about the CADs and facility comparison, and you were a little concerned about the different sampling rates...

A: Again, that is just for a small time slice.

Q: We were looking at it not from an engine anomaly standpoint, we are still trying to figure out if we should screen some of this stuff (or if we ought to screen some of the this stuff) before any of the other models look at it. Because if there is a gross difference between a CADs and facility measurement, maybe we could eliminate one of those, so instead of looking at 754 they can use the CADs speed measurement.

A: Yeah, there won't be any real gross differences, maybe 40 psi in mainstage up, and you've got to have some sort of logic gives you some leeway during the transient. One way that the controller handles that for main chamber qualification is that it allows a delta between the commanded pc and the measured pc of 200 psi whenever the command is changing. It knows when the changing the command it looks at the previous cycle and if its a different command it puts the delta limit at 200. And then it says when the command is not changed or 50 major cycles (which would be for one second) then I change the limit to 75 psi. You're going to have something similar to that in a lot of your rules.

Q: Doesn't it do that also to the turbine discharge temps, A minus B?

A: No, that is two step redlines...

Q: What does a two step redline mean?

A: A lot of parameters have redlines that are different for different power levels. There are only five or six parameters that have redlines, but there is a start redline on the turbine discharge temp that is lower, this only lasts for a brief period of time. That because we always start at 100% power level. The logic here is you want to make sure that it's cold enough at 100% so that after you lift off and you go to the higher power level that you have a margin to your real redline. That's not really sensor validation.

There is a difference in the facility and CADs as far as (I can't get into the electronics or the way the data is handled), but there are filters that cause the data characteristics in a real transient condition to look a lot different. I don't know if that's anything you guys will have to worry about. We'll get into a major incident where you have a huge change in a very short amount of time. We had one where we boiled off the lox pump and the ? speed went from 26 to 44 thousand rpm in ten (mils or minutes ?). The difference is, I think the cads data had characteristics like "this" and the facility data was more like "that" (drawing things). We did different some sample rates, they are filtered differently is all I'm trying to say is that I don't know how you account for that in a changing condition. For most normal changes transients doesn't matter. It all comes down that you have to know the characteristics of instrumentation. We've had two different types of temperature sensors in the turbine discharge, the start transient, the RTD, the resistance ? would do more like "this", during the one second, and with thermocouple in, there would be a response like "that". If we saw it on an RTD it would be a big problem.

Q: Do the use RTDs all the time now?

A3-195

A: Yeah.

Q: When did this take place?

A: We were switching back and forth. On test data you're going to get some thermocouple.

Q: This may be important if those two tests were being used as the comparison tests. This may decide whether or not you had a thermocouple or RTD, but it still would need to be flagged up front.

A: That's the point, as long as it says that we had a significant change in this parameter at this time, along with a footnote which said that we went from a RTD to a thermocouple, then the engineer would see immediately why the change is there.

Another example, though, of something that is a sensor validation problem is that we have two different brands of temperature transducers (ccv's and Stathan?). They have a 20 psi (I think) delta. One of them is not allowed for flight engine redline parameters, but the other is. But, you'll see a pressure (mainly the intermediate seal purge pressure) that is close to the min. limit during prestart _____... {can't hear}.

Q: {can't hear the question}

A: Tracer. It's the engine configuration.

Q: You mean that you can only tell from tracer? Would it show up in the pretest package, like in h/ware changes?

A: It should show up in the engine configuration portion of the original pretest (package). But, I'm not sure.

{Conversation about the tracer transfer problem}

Q: I understood (about Tracer) that the information didn't get into it in a timely enough manner.

A: Right now it's not being used in that (time crunch) type of mode.

Q: Maybe if something's off always by 20 psi, you could just assume, and post a question?

A: I think that that's a good idea. Postulate and ask the data analysts to try to verify. That's what we do.

Q: Would it be good to go through each of the data books and look at every single plot or every single transducer or list of the cads pids and facility pids and see that if this one exhibits this type of historical behavior we could switch this out? I'm trying to find out what the best way is to go about getting this.

{Asking how to best go about getting our needed sensor validation info}

A: The last part is the tricky part...I don't know how methodical it would be, I've looked at 5-600 tests before that. You need data that exhibits nominal behavior routinely, or pretty often.

Q: What would be nice in this case, is data that exhibits sensor problems so we know, well you were able to give quite a few example as you were going through, but the only way, we need to get as much of that as possible.

A: Maybe going through a pid list would be the best way to it. You should be able to get the zero shifts from the model people.

For now, I'll just tell you what I can think of:

The bleed valves have LVDT's that have, {two different lots...} {can't hear}... the cryogenics and when the bleed valve

closes at 100% and it tappers off and it does something like this and it gradually closes. It starts to close at start enable, at 2.3 seconds engine start and at start enable plus 2 sec., the controller checks it to be below 20 degrees and one of the lots plateau out right at 20, in fact we can cut it off right on an FRS pad. So they disallow use of those on the fuel side and they only use them on the lox side, because the lox side is warmer than the fuel side, about 120 degree. We are using one of these out on test bed, and we just raised the limit to 30% max position. You would see 10-15 percent delta in the position, depending on which LVDT you use. It is important in that you have two different levels depending on what sensor you use. That may just be a limit check.

Q: You said this was done by different lots?

A: A certain lot of them had displayed the different characteristics.

Q: I'm confused, is the lot all gone now, you don't have that problem anymore?

A: They still exist, they built certain bleed valves, and those bleed valves come out use a certain lot of LVDTs and they are in the program. And we try to separate them and use those on the lox side for flight engines. And what we do on the test end is address the concern, we can raise the limit or we can take our chances. You wouldn't want your program or model to say there was a problem because it was close to 20 if it knew it had one of those LVDTs.

Q: So, what your saying is that at start enable plus 2 seconds for this parameter, there is a check below 20, but on all tests you could say its below 30 and be okay, it just flight that absolutely needs to be below 20?

A: Well, the limit's in there all the time. You're checking against the limit, your program will check against the hard limit and will probably also look for changes, and, the level that it settles down at, even if it is below the limit. The limit is the last, the hard limit. I guess I see what you guys are doing, you have data here and your checking here against what it's expected to be at, here is your historical two sigma, type bands, bracketing those or hard limits. (Scribbling wildly) Maybe layers of checks, with maybe in "here" a tight band that calls out a potential problem. What the model has to do is tell us that we are moving towards a situation.

An example would be where we had a purge pressure (we purge with nitrogen) until some time past purge sequence number four, it's a nitrogen purge in the preburner, we turn the purge off and check that to be below 50 psi, that's the 50 psi max limit. When you cut it off and met other requirements you get engine ready, then you get engine start. We've got a violation in this limit, at engine start on one test it went like this (drawing) and broke the 50 psi limit, we went back and looked at the test and instead of coming down here to 14.7 it was hanging up around here and gradually going down, which told us we had a leak in a check valve that we didn't catch because it was well within these limits. We then started looking at that parameter to make sure it was going down "here". I assume what you guys are going to do is check against the numbers of different criteria and have various levels of concern.

You could have a similar situation if a sensor is going bad and you still want to qualify it. You'll want a tighter check on it to give you an indication that it is going bad before it actually goes bad.

Going back to the bleed value, if I had one of the bleed valves that was supposed to be at 6 and it was reading 20, then that wouldn't be good. I couldn't say that that was OK.

Q: How do you know that it was supposed to read 6?

A: You have two different databases. (I think he's trying to

A3-197

explain that it's in his head, this part is hard to hear}

The MCC coolant discharge pressure, PID #17: I think that the discharge pressure and temps are here (PIDs 17, 18?) and it goes down and goes into the low pressure turbine inlet. Before it goes in here you have to get 436, which is pressure. If you plot them both together, 436 might be like "that" {drawing things} and 17 is a higher pressure. There is one example of something where you can go back and compare another parameter in the system to validate.

Q: Does it normally drift because there is a sensor problem there?

A: I assume that it's a sensor problem. Low parameters: I don't put a lot of weight on them.

Q: Really if 17 was moving around and 436 wasn't you should be pretty suspicious?

A: That's right. It doesn't necessarily have to be the sensor, but at that point I would assume it. The reason I say that is if you look at the hot gas manifold pressure (and that parameter just goes all over), they have an acrylic vinyl power head and they can introduce bubbles into it so you can see the flow. The exhaust from the fuel turbine goes all the way to the power head(?).

Q: Are they working now with 2 ducts?

A: They're running them.

Q: Weren't they eating up the chambers?

A: Well, I'm not sure if they've resolved that yet or not, but that's what they're going to be flying.

{Commentary on the 2 duct stuff}

Anyway, you have your injectors here, and the flow from the fuel side is coming all the way over here. I have always been told that the stagnation point is something like this, it probably varies with power levels. My point is, any pressure taken in here is subject to the flow dynamics in a very complex system. You just have to understand what the physical conditions are that reach instrumentation.

{End of Tape4, SV#1}

{Conversation with Marc Neely continued on Tape5, Neely/Foust}

{Marc Neely continued}

{Tape marked tape 5 Neely/Dave Foust}

Q: Is it because this it's unclear where the stagnation point is that the hot gas manifold press jumps all over the place?

A: Yes. Depending on the power level and the balance between the pumps, the pump efficiencies can cause a difference in this. You couldn't have a very high efficiency fuel turbine and a low efficiency lox turbine that requires more hot gas flow to drive it to maintain the mixture ratio. This changes that stagnation point. If the stagnation point is right on the transducer then...

That's not a sensor prob, that's a question of how valid the reading is.

The pump speeds, mostly the low press pump but sometimes the high press pump, they tend to have a problem. At times they display a real wide band of data.

Q: Pid 30 does this often.

A: There is a probe that reads something on the shaft (a magnetic

reading) as it's rotating. If the probe gets too close to the (shaft) and the shaft has some axial or radial movement, then the signal can be distorted.

{can't hear}

Q: There you'd check for excessive noise? Typically there's some fluctuation but not on the order of 1000 rpm?

A: Yes.

{June asking something}

A: We don't look at a speed that's real erratic. We don't look at the pump discharge press, for example, we just assume there is something wrong with that parameter.

Q: The sensor is really not bad there. Don't you use the average as the correct value?

A: You could. Maybe you'd need a correction.

Q: Does the thing go back to normal?

A: It can, typically it doesn't. If it's bad, it's bad.

There's a similar problem with the fuel flowmeter speed where there's a synchronous frequency oscillation in the data. This is a function of the sampling rate vs. the controller cycle. I wouldn't worry about that. It has an effect on the engine performance, but it's very, very slight. It could cause valve position commands to "do that" {drawing} but not to a magnitude such that it affects the actual (valve) position.

Q: If the fuel flowmeter is calibrated and Kf is off, when you plot out the actual parameter, do you know what those pids are for and how we can use them to validate (Kf)?

A: The model people may be the better people to ask. The Kf pid is just a constant, I don't know what plotting it does for you.

Q: Well, it definitely varies with power level.

A: The Kf is just a constant that you use in the calculation of volumetric flow that says if the flowmeter turns at this speed, then the volumetric flow is whatever.

The only way to make sure that flowrate is correct is to do data reduction. That's the primary function of the modelling group: to validate Kf and C2. They have a process to do that, it gets to be very small adjustments, like exit plane pressure. It's all ISP.

I can't think of any other instrumentation that you'd have a problem with. Other than failure characteristics.

Q: Failure characteristics of the sensors?

A: No, of the engine.

{Long conversation on who we're interviewing during the week}

{Question not recorded}

As long as you have hard limits in, I think you'll be ok. The limits that the controller as, is for failure. Anything outside those limits is a failure. You may want less stringent limits. We've had situations where the data changes slightly over a series of tests, without a significant change from one test to its previous test. You have to catch these kinds of trends.

Q: Wouldn't that show up in the 2-sigma comparison?

A: Your 2-sigma is going to tend to grow. We just apply 2-sigma

A3-199

database to the max lox turbine temp condition and max fuel turbine temp condition, and those are generally opposite. They take place at different times. If you don't have similar test profiles, it's very hard to construct a 2-sigma database. I don't know how you'd do it in real time.

Q: Right now the data package contains certain start and shutdown 2-sigma comparisons.

A: But it's only valid during certain time frames.

Q: You mentioned before that if one of those bleed valves was reading 6% for several test firings and then all of a sudden it read 19%, it's still below the limit, but you'd be suspicious if it's the same valve and nothing else has been changed out.

A: Right.

Q: Can you break up this type of reasoning by component?

A: I'd break it up per parameter.

Q: Aren't there several measurements that indicate directly the health of a particular component?

A: All of them ought to.

Q: Do the modelling people trend, say, pump efficiency?

A: Yes, but there's a lot of leniency on that. There shouldn't be much leniency at all on ... I don't know how you can put importance on that.

{Catherine is ordering pizza}

I have limits in my head, bands, that I reason with.

Q: Is this a delta from another test?

A: No, these bands are built up of my experience, what I've seen before, what I know from reading the pretest, what I expect, and these are very tight bands because a data analyst must be conservative. You don't want to think it's ok when it's not. Then you have these bands (may be 2-sigma bands) which may be looser. Then you have the hard limits (controller limits?). You're probably looking for something that, for instance, doesn't matter if it's not nominal if we expected it. I don't know how you'd do that except on a parameter by parameter level.

Q: Intelligent limits.

Q: The High Pot system has a table of tolerances based on the experts' views of what to expect. They are also tied to the statistics (mean, standard deviation). The idea would be to update this table based on any new information.

A: You're going to compare what's expected to what's taking place?

{We flag things that are outside of a delta compared to previous tests. Starting with the tightest delta}

You can compare two anomalous test, they can both be the same and still both be wrong. We had a situation where we turn off the purge, and have a 50 psi limit which is supposed to come down to 14.7. Over a series of tests it came down and stayed up a little high, gradually tailoring off at engine start. On one test it came in below 50 psi and on engine start, it started up. This was a check valve that gradually leaked more and more and eventually cut the test. You have to be careful about this type of thing. I hope we're not looking to see if it looks exactly like it did last time.

{Defensive action by Tim}

The other thing you'll run into is if there aren't comparison tests. (i.e. the first run of an engine, or new pumps, valves...)

{Defensive action by Claudia}

{Jean explains phases}

Then you miss anomalies.

{Jean explains events}

Even if I'm looking at 100% rpl, 30 npsp on the lox, 8 on the fuel, and that's a phase with nominal repress conditions, I may not have a similar point or similar phase on a comparison test. Then I'd have to apply gains, that gets subjective.

{End of conversation with Marc Neely}

Begins middle of tape marked "5" in Claudia's collection

{Amy is currently working on this. She will mail out a revised version as soon as she can figure out what plot packages Mr. Foust is referring to, and mark the pages in the text. This is anticipated to take a few days.}

4-30-92 conversation with David Foust about sensor validation, phases, and features. Going over data packages from test 902-551, engine 2017.

A: This is the calibration test for quad engine 2017, it does have venting, is a flight engine, and has a couple green run pumps on it. I'll point out what's normal in terms of instrumentation and point out ranges of what we look for as what's normal or abnormal.

{Looking through ??? package}

This is the pcv actuator position A & B. There's a 3/4 of a percent delta between A & B channels. That's a little bit high. Usually it's less than about 1/2 of a percent.

FPB pc: (NFD is "non-flight data" that means its a facility measurement) it drifts and is erratic during mainstage. We tend to see FPB pc drift. This is pretty common. It being erratic - that means that the cable is bad or something along those lines. OPB pc goes bad at start plus 236 seconds. PB pump discharge pressure (NFD) drops 120 psi from engine start plus 95 to 125 seconds. The hpop discharge pressure NFD shifts and spikes throughout mainstage. The hpop primary seal drain pressure #3 (there's three measurements on that) is high/erratic throughout the test. HPFP disch pressure channel A is low from engine start plus 5 to 63 sec and then **** facility PID.

{Looking through instrumentation stuff, don't know what package yet}

CCV actuator position (page 55) has a delta early in the test, before the bucket which is fairly small and starts growing throughout the test. You can see it in the 3g throttle and when they cut off. I don't think it was the magnitude so much as they were calling out that it's zero here, and came to be a delta later. This is kind of unusual.

Q: So, a certain delta could be expected but you would expect it to be the same throughout the test?

A: Yes. The delta is not really all that big on this particular plot. We've seen it as much as 1 percent between the A and the B. But, the fact that it was zero here and have a delta of 3/4% to a percent out here, that's kind of unusual. Typically it will stay constant throughout pre-start, mainstage, and shutdown.

Q: Unusual for any actuator position?

A: Yes. They should stay fairly constant if there's a delta between them.

Next one, this is a calculation of the delta. A minus B.

Here is fpb chamber pressure. This is a typical measurement where we see drift a lot. I mean by drift that it tends to, say, start up out here at say 104% and by the end of the test at 104% you might be 50-150 psi lower. (He points to erratic behavior in the data and says that this is not a drift). It's a thermal drift that they had a fix for called a remote mount transducer. Typically, the pressure transducer is right there where the pressure measurement is. They put a little sense line that remote mounts the transducer from where the pressure is measured. This

A3-202

way you don't get the big thermal effect on the transducer itself.

Q: Do you not see this in current tests?

A: You still do because they haven't gotten them on all engines yet. There are still flight and ground test engines that don't have them yet. But, if you see it you know it's thermal drift.

Q: You see this only during mainstage?

A: Typically, yes. Because that's when your propellants are flowing and you're nice and cold right there.

Q: And then you would see the signal recover?

A: You would see it recover during post-shutdown. You're still pretty cold after test. We pretty much don't look at it after shutdown, but it's going to warm back up.

Q: When you see this drift, do you actually find what the configuration is for that sensor (new or not)?

A: Typically, we'll just ask "them" during the data reviews if this was remote mount or not.

Q: If that were remote mount and yo saw this, would you be worried?

A: It still wouldn't be that big a concern. We actually have seen a couple of tests, that are kind of outliers, that have it. They don't have a consistent way of doing remote mount. Sometimes you'll have alot of special instrumentation and you can't do it the same way.

Q: The drift is only for PID 410, not PID 158?

A: Right. This one's more than drift. It's actually a little bit erratic {looking at a plot}. A drift has a smoother characteristic, a slow, gradual drop. I wouldn't expect any kind of "bumping around" like this.

Q: You just expect 158 with the slope?

A: Right. a very slight slope on it.

The fact that they're right on top of one another, and starting to get a delta in the bucket...that delta gets pretty big (about 250 psi). I've seen as much as about 150 psi on a drift and that's pretty high. So, this is more than just a drift. You can see that the CAD's measurement is flat - so that's probably a remote mount. The facility and CADs are both off the same transducer, they just tee off each line. The drift is common in flight also, if their isn't a remote mount.

This FPB pc. They have three different tests here and are showing the trend. The main thing is that it's now negative. Obviously, then, the sensor is off. It's significantly lower, like a constant delta off. They probably had a vacuum reference leak or something along those lines. We can check the next pre-test to see what they actually found.

{Don't understand what we were talking about here. May want to ignore this section}

Q: Does that partially explain that 250 psi on the previous graph?

A: Right

That's just showing it post-shutdown, saying that we saw it

erratic here in mainstage, looked good in start, I don't know what it looked like in pre-start, but in post-shutdown and it's definitely because you can't have a negative pressure.

Q: But it's still up at post-shutdown, then that wasn't thermal drift, right?

A: No, I don't think it was drift. On this one {flips pages} the CADS measurement is flat, so I think this one had a remote mount. I think this is instrumentation. It was not thermal drift.

{flips to another graph and points out...} That's just a confirmation that it was bad and stayed bad after shutdown.

LOX PB pc: don't really see drift on this, see it react to the vent a little. You can see the delta, then it starts to go erratic on the 3g throttle.

Q: What is an acceptable delta?

A: They should be pretty close. This is a little on the high side.

Q: Why did you say that it was reacting to the vent?

A: The OPB pc tends to react. The lox turbine temps really show how the lox side reacts to the vents. Here's the lox vent. they vent down and then repressurize the system. {looking at lox turbine temps now} See how they're increasing and then start to decrease. This is a Power level change. Then they continue to decrease. That's just a reaction to the vent.

Q: This is comparing the same PID's from 2 different tests. So, is this delta still acceptable?

A: I'd say there's different pumps or engines. That explains alot of that. So, you're right, they shouldn't be right on top of each other. The spread being called out as instrumentation occurs near the 3g throttle.

Q: Without looking at any other parameters, you called this out as instrumentation problems?

A: Yes. The 3g throttle is a nice, 5% per second throttle down and is real consistent. You can see it stepping down normally here {pointing at graph}. In this you see it doing something kind of unusual. We don't have a CADS measurement on OPBP.

{looking at another graph}

This is facility meas. vs. CADS. You see here that the facility was bad and CADS was good.

Q: What causes this to drop and then come back up?

A: Could be a loose cable on the connector to the sensor...

Q: Do you still use this measurement in "these" ranges?

A: I'd say that it's probably still pretty good. If I had a choice I'd pick the CADs measurement because it's nice and steady. You obviously don't have a power level change in this area so you know it's not real.

Q: Would you first make sure there weren't some other shifts going on?

A: Sure. That's the first thing you check: Do I have some power level shift or repress change or something unusual. Typically on a test like this, we're going to repress out here around 200 seconds {may have been saying we won't typically...}.

Q: But why would one see it and not the other, if they're

A3-205

measuring the same thing?

A: You tee one wire off the pressure transducer to the facility measurement system, and one wire off to the CADs measurement system. You may have a loose wire, cable shaking, water in the sense line going to the facility,...

Q: If you saw one jump down, but didn't see the other one jump down, would you still track it and see if there was a power level shift?

A: Sure.

Q: So, really, you don't know if this sensor is bad, or the other one at first?

A: Sure, anytime you see two measurements that are different, you don't know which is bad. Now, I know a calibration test profile and I know that there's no (power level change). I know that it's the facility measurement that's bad. If you didn't know that for sure you'd have to see if they did a power level change, did something else go on, a repress change, something else in the facility, you would investigate other parameters in that time frame. If you don't see anything else moving, then you can say that the facility measurement has the problem.

On the ground we usually have the CADs vs. facility on a lot of measurements which gives us a good way to check out what's bad.

Q: Do you have normal delta's anywhere stored for CADs vs. facility?

A: No, there's nothing really hard and fast written down.

Q: How far would be too far apart?

A: I would say, typically on the OPB or FPB (preburner pump), you're talking about a transducer that's probably about an 8000 pound transducer. The accuracy is probably about 15 psi. If it's much more than 25 psi, then I would probably start looking at it a little closer. But even on that, it's a pretty big transducer, this comes into engineering judgement.

This is the HPOP disch. pressure CADs vs. facility and you see here that the CAD is good and the facility is a little bit erratic.

Q: Is there something wrong with the facility data collection here?

A: It may be they have some kind of problem with the facility. This is unusual to have this many facility measurements going bad.

Once again, you see if you had any power level changes. Here you see a delta here where you didn't see a delta early on. You can double check but it looks like the facility measurement getting erratic.

Q: This small rise here? They're both doing it...

A: Then it's probably real.

Q: Would you check that? Does it mean anything to you?

A: On a scale of 250 psi, this is maybe 50. That's maybe a little on the CADs but on facility it's only about 25 psi.

Q: So, you actually think about what your transducer's capabilities are when you go through this?

A: Yes, you have to be careful because sometimes the scale is really tight and something may look enormous when it is really tiny. The HPOP disch. pressure is about 5000 pounds, pretty good sized.

Q: Do you have a list of the sensitivities and ranges?

A: Probably the PID list lists this.

Q: If you saw chamber pressures and other parameters rising by the same types of percentages, would you look for an anomaly?

A: If I saw everything in the system reacting, I'd have to look at it a little bit. You're going to see a little bit of bumping around. HPOP disch. pressure is typically really flat. If you see any kind of bumps or drifts in the HPOP disch. pressure, that's unusual. That's what the engine controls to and it does not change alot in one engine from test to test. This engine is running at 104% at 4025 psi, you can consistently expect that engine to run at that pressure, +/- about 10-15 psi. If it changes, then there's something unusual going on. We try to anchor our predictions on HPOP disch. pressure because it's consistent.

This is the HPOP primary seal drain pressure. You have 3 pressure measurements, p1 p2 and p3. Typically, they are right on top of each other. This is a small scale. This is gauge pressure. I've seen them spread as much as maybe 0.2 psi apart and that wouldn't bother me. The thing that caught there eye is that it's erratic.. "this guy is bouncing around, and shifting" which tells you there's something odd going on in that one measurement. They're all in the drain line so they all should be pretty consistent.

Q: Are these three different sensors?

A: I believe these are three separate transducers.

Q: Is there a delta that you'd worry about?

A: I'd say if it got up to around a half psi I'd be concerned.

{End of Tape}

{David Foust 3 side A: David Foust 2 had nothing on it!}

A: So, I've seen the deltas, I would say if I say one here at -.4, -.2 and zero, that would be a pretty good size spread but it's a pretty small pressure difference, so I wouldn't be too concerned about it. You might mention it to them and recommend that they check it out, but Stennis is not going to be able to do alot. I'd maybe call it out, but if I saw about .5 psi (delta) I'd start getting a little concerned. Now, if all 3 of them were high (3 or 4 psi) then you'd probably have a slight leak down that drain.

Q: Both these traces are drifting around in time. Is that a real engine phenomenon?

A: Yeah, that's kind of a response to the bucket. You can see the power level response. It also responds to the LOX vent. WE peaked out on min. vent and are repressurizing in "this area here." The whole LOX side responds to a LOX vent - even the fuel turbine temps.

Q: So if this was really consistent here, a constant line but shifted up, would you have a problem?

A: It depends on how big the shift was. If it's about .2, that's not too bad. If it was "up here" then you might expect a scaling constant on the transducer to be off.

Q: How closely would this trace follow "it". Would this bother you where the delta is changing slightly?

A: That's a little unusual. Typically the drain line traces are pretty consistent. That's a pretty small drain line so the 3 pressures should be doing about the same thing.

A3-207

Q: Is this one of the parameters where you could confirm if there's a leak by looking at prestart?

A: Yes. You would see it leak.

Q: So, you would go back to prestart and check it?

A: Chances are that you're not going to see anything in prestart because you're not flowing anything through the pump. But if you were leaking through there and it did get in there, you'd see it in prestart. The LOX system is pressurized to about 110 psi, which would be enough to push it through the drain. The drain temps are the key. If they drop like a rock, then you know you have a leak.

This is CADs vs. facility HPOP disch. pressure. It looks like it's drifting. You have a delta here, but not out here. It's low from engine start plus 5-63 seconds *** degrees. The fact that it drifts a little is not too unusual, although typically it is fairly consistent. This is a fairly small scale, it's only drifting 25-30 psi. That's not too bad.

Q: Over what time are you saying this?

A: From here to here. You're venting during that time, at 109% so that's not too unusual to see a little bit of a drift in that measurement. The main thing is that you have a delta during start, right after start, before the bucket and even during the bucket. Once you come out of the bucket you're delta goes away. This is what they're calling out.

Fuel vent doesn't have as much of an effect on everything like LOX venting does. The mainstage is compared to 537, another calibration test with the same duration and same power level profile.

Q: Would they have thought that "this" delta is too wide, because "this one" is right on.

A: Not really, these are valve positions. FPOV A and B side, for each test compared against each other. You have to be careful with the scale. This is maybe 0.5% which is getting kind of high between the A and B channel on FPOV. It is consistent throughout the test, but is a little bit large of a delta. The fact that one test is here and the other here doesn't concern me because of different engines, different pumps. You would expect that.

Q: Does the high delta between the A and B channels on one test concern you about the engine or the instrumentation?

A: The instrumentation because it's consistent.

Q: So if you have a constant delta, you're not concerned unless the delta's too high. And if it's too high you suspect instrumentation?

A: Maybe instrumentation, maybe the engine is running funny. You have to check everything out. Typically when you have an instr. problem, you check prestart. If you have a delta in mainstage, did you have it in prestart? post-shutdown, did it continue through shutdown and post.

If it didn't, check other things. For example if we suspected FPOV, did we see the FPB pc spike at that time? Did FP speed go up at that time? If it's a real valve shift we would expect a reaction throughout the system, if it's instr., then nothing would react. It's nice if you have CADs vs. facility but this is a CADs only measurement.

Q: Let's say you called out that 0.5 % delta, I'm assuming the modelling group uses alot of that data. Do you know which one they would use?

A: Typically, we always use the A channel. If there's a question, we check each vs. FPOV commanded value. We always have the command to fall back on.

{end of this package, start with next}

{**started marking pages here**}

{Flipping through package marked "pretest" with no cover in our set}

This is pretest, they give thrust profile, h/ware changes, etc., times where we have min. lox vents, max lox repress, fuel vents, doesn't show a repress change on here. The fuel has a min. max repress cycle.

Q: The lox doesn't have repress anymore?

A: The lox has a fixed orifice on flight now.

The pretest comes in real handy, we also have the test request. Some things are in here that aren't in the pretest.

{End of pretest package stuff}

{Going over GRPUMPS high pressure fuel pump package}

{calc 1}

This is for HPFP, most the parameters in this package are geared toward the HPFP. This is FP disch. temp minus FP inlet temp (delta temp. across HPFP). We look for any kind of shifts other than power level shifts.

Q: Would that indicate an efficiency change?

A: Could be, also could be some kind of leakage. Could be instr.

Q: What would you consider a shift?

A: A shift up and down, or just shift up. We've seen that a few times.

Q: More than 2 standard deviations?

A: Anything you'd catch by eye. You're not changing power level or doing anything at that time, so you'd catch any change. On the average you'd expect a delta temp of between 48 and 50. That's an average pump.

{calc 2}

This is FP discharge pressure minus balance cavity pressure. The balance cavity pressure is an indication of axial rotor motion. Looking for shifts. You'll see a little drift, that's not unusual.

Q: The drift {downward slope from 65 sec to 170 sec} is consistent from test to test?

A: I don't look at it too often from test to test. I would think it would be consistent.

Q: Does it always drift down, and then up when you're going up {upward slope from 1 to 60 seconds}?

A: This is start overshoot. You'll see a lot of things in the first 15 or 20 sec's that you won't see anywhere else. That's just thermal transient, start overshoot... This may be reacting a little to the fuel vent. A downward drift is in the right direction.

A3-209

Q: You can expect a downward drift on the order of 50?

A: This is typical.

Q: Any spikes?

A: It would be more of a shift. Sometimes a sine wave, that would be something bouncing around. The lox pump has alot more anomalies (balance cavity-wise) than the fuel pump does.

Q: It's still drifting upward in the bucket.

A: You're going from 104% to 65%, that's a big change in power level and thermals. Then you only stay there for 10-12 seconds. That's not unusual.

You're running on mixture ratio control, so everything is normal around 104%, 109%, at 65% the engine is not running peak performance. You might see some unusual things like B channel higher than A channel in turbine temps here, then they flip-flop in the bucket.

Q: Do they flip up again when they go back up?

A: Yes. Occasionally the lox, but usually we see the flip-flop in the fuel. The fuel is more sensitive. The swirl coming out changes because of powering down the turbine and the coolant flow may react differently. You may have coolant hit a sensor...

{calc 32 or 3?}

This is coolant liner pressure. It's mostly a steady pressure. Looking not to exceed redline, make sure the pressure stays high enough to get enough coolant in the fuel pump. Looking for erratic motion that might indicate ice build-up, etc.

Q: Is this the measurement that ices alot?

A: Yes.

Q: Do you look for erratic motion or a flat line?

A: If it totally got blocked, it wouldn't respond to a power level.

Q: Do these tend to follow each other closely?

A: This is a red line, so this is calculated.

Q: Aren't there A and B both here?

A: Yes. That's one pressure transducer with a bridge so they should be pretty close.

Q: So, if you saw that they were drifting you'd be concerned?

A: Yes. If I saw a good size delta or a drift in one, I'd be concerned.

Q: On the icing, is that and instr. prob or ice forming in the line?

A: That would be an anomaly.

Q: Does it mean much?

A: Well, it means you're blocking coolant flow? Could be icing in the line or in the actual coolant flow. The main concern of this parameter is the coolant flow, make sure you're not getting a reduction in coolant flow that could overheat the turbine and bearings, etc...

{I think responding to the fact that there's is or isn't a lower

limit on the parameter marked on the chart}

There's probably a sensor qualification limit saying that if it reads so low, the transducer's probably messed up. That would be the lower limit.

Q: When you say erratic motion again, you're talking about a wave?

A: A wave or a step.

{June's drawing things on a graph and he's commenting whether or not the variations are normal. See plot package}

{#4}

This is coolant liner temp. This is also an indicator if you have a problem. {looking through pretest package} From system's level, we just say we saw something, and Glenn would analyze it. Ask Glenn about pump limits.

Q: Is "that" spike normal {10 seconds, 20 degree spike}?

A: That's start transient, yes.

Q: What would be abnormal?

A: Looking for erratic temperatures, shift down and back up, and overall level, 300-310 at 104% is about normal. If you got really hot or cold, then that's an indication that something is going on. Typically look for cold.

Q: Average is like 280-320? Of tighter than that?

A: I'd have to ask Glenn. See reacting to vent here as well.

Q: Who looks at it close?

A: Typically Glenn.

{#5}

This is FP drain pressure, looking for low pressure indicating no leaks. On A1 they don't have a diffuser so you'd expect a higher pressure on A2 and B1 they do which drop the press's down to several hundred thousand feet altitude. Have to be careful comparing A1 to other stands. A1 would be around 14.7. Sometimes the stands will read psi, or psig.

{#6}

Here's drain temperature {HPFP}. A leak would be a drop of several hundred degrees.

Q: {pointing at plot, can't tell what at} Is "this" normal?

A: Yes, just looking for levels.

{#7}

This is lox pump bal. cav. pressure. There's a 1a and 2a. Looking for rotor motion. One thing we see fairly common is one channel shift up but the other channel will not move at all. This is not real rotor motion. You'd expect to see one to go up and the other go in the opposite direction for real rotor motion.

Q: When it's not real rotor motion, is it instr?

A: Probably, have to talk to Glenn. I think it's some kind of seal leakage.

Q: What if they both shift in the same direction?

A3-211

A: I've not seen that, if I did I'd probably go talk to Glenn about it.

I would flag one shifting as an observation.

Q: The deltas are changing, does that concern you (180 to 230 seconds, upward shift in delta)?

A: This one's coming right out of a power shift, that is a little unusual. It tends to be bigger down in the bucket, that's typical. The 1A channel here is dragging, I might bring that to Glenn's attention. Every once in a while you see one drag (react slower). It may be the rotor moving slower, etc. You expect the rotor to change a little on a power level shift. You are speeding things up, putting more power to the turbine, and will push it towards the pump end.

Q: Is the fact that when you're in the bucket and the delta doesn't concern you hold true for most pressures?

A: I know it's typical for HPOP bal. cav., I'll have to look as we go.

Q: Do you look at the fuel side as a delta?

A: You only have one bal cav measurement on the fuel side so you look at the FP disch pressure minus bal cav pressure. Also look at 1A minus 2A. Then the shift will be just a straight shift.

Q: What about for sensor problems?

A: Look for a spike or any type of erratic behavior.

{#8}

This is HPOP inter. seal purge pressure (redline). This is a flight LCC. Basically, look for on mainstage, erratic motion in the nature of coming up, then you see some kind of wiggle and it straightens back out. That indicates some kind of rubbing of the seal. This thing varies alot during mainstage, I've seen some that had a nice dome shape and some that come up and almost be straight and flat. Really, you are just looking for is an erratic wiggle or bump. Prestart, you look at levels. The flight limit is 175 (?) at engine start and when the purges come on which take some of the Helium away, it will drop down to 170.

Q: If you got the erratic behavior, but all the time, would you suspect the instr?

A: If it was consistent, I would think it was instr. but we'd have to check it out.

Q: How would you do that?

A: It's hard. Sometimes if it's significant rubbing, you might see it in the dynamic data (accelerometers on the pump). If it's minor, you never know for sure. Might see an indication down the drain line, in the form of significant pressure shifts or oscillations.

{#9}

This is turbine secondary seal cavity pressure. Looking for a nice low pressure. If the turbines run at a high press, that's after 2 seals and means you have a pretty good seal leakage. (pointing to 10 to 40 sec spike) This is the typical start spike, then comes down in response to the bucket. That's a little bit low, spike-wise. Glenn looks at the spike level, and the overall level.

Q: What's important about the spike level?

A: It's just a characteristic we monitor, I don't know if it's

significant. We always see a start overshoot in this parameter. Typically it's around 28-30.

Q: This spike "here" doesn't mean anything? (pointing to spikes within the overall start overshoot)

A: That's a little unusual. I wouldn't be too concerned about it. I might point it out if I saw any kind of spiky trend in the rest of the data.

Q: Do you look for the rubbing trend in this parameter also?

A: Not so much rubbing, basically the overall pressure level in that cavity.

Q: Do they normally lie so close on top of each other so you'd get worried if the delta was high?

A: They're pretty close. It's a pretty small scale. If they were much more than 1 psi, I might call that out. Once again, it's one sensor with bridge. They're usually right on top of each other.

Q: When you say small scale, are you referencing this to the range on the sensor?

A: No, just talking about a few psi. Have to pay attention to the scale.

Q: If the parameter went way above "this" would you be concerned?

A: Yes, if it did then that would be a real concern.

{10}

This is the secondary seal drain temp. Don't want this too high. If you're leaking around the turbine seals into this cavity, the turbines are running at 1300-1400 R, it would be hot. You expect a decent temp. drop by the time you get to the secondary turbine seal. Don't want to be too hot.

Q: When comparing to a nominal test, what would be high?

A: If it got up to 950-1000 degrees. On an old pump, you'd expect it to be a little high because it's worn.

Q: Why don't you plot against past tests?

A: Don't know. Glenn keeps track of levels in his head over tests.

Q: Does he make recommendation about pump condition?

A: Yes.

Q: This side has an increase in variation (after 200 seconds) does that bother you? Looks like it's maybe 5 or so times bigger than the peak-to-peak oscillations here?

A: I wouldn't call that out.

Q: Is it typical to get some more high frequency stuff toward the end?

A: Going through 3g throttle during this time, then a big power level shift. You're a lot more thermally stabilized toward the end. May be thermal effect, may be vent effect.

{#11}

This is primary turbine seal drain pressure. Same thing.

A3-213

This is a typical start overshoot {0 to 20 seconds spike}. Once you're in steady state, you're in the 15 psi range. That's about where you'd want to be. Looking for a higher overall pressure indicating a leak.

Q: If you saw a little shift upward, would you worry?

A: A slope change? Yes, that would catch my eye.

Q: How repeatable are these oscillations.

A: I'd expect anything within a certain amount. If I had a big overshoot, that would be unusual. The drift down is typical.

{#12}

This is the same thing, but for temperature. The level is higher than the secondary temp was. This is the drain line temp. Basically, look for this to get really hot. Right after primary seal so if you had a leak from

turbine, it would be several hundred degrees higher. 900 is about where we expect it.

Q: Do you expect temperature sensors to react slower, so if you saw high frequency oscillations, you'd suspect the instrumentation?

A: The temps are pretty slow to respond. The pressures aren't that high frequency either.

Q: Do you ever decide that a spike couldn't be real because the sensor is unable to pick up a change that fast?

A: We could think along those lines. Also only sampling at 25 or 50 samples per second.

{calc 3 or 13}

This is HPOT disch. temp minus primary seal temp. (delta t across primary seal). Looking for levels.

Q: What levels are too low or too high?

A: If the delta was zero, you have no seal. Want to be in the 400-500 range to be a good seal. Typically see a higher delta t on higher turbine temps. If the turbine temps are cold, running at 1300 degrees, the delta t would be lower on this plot than for the case where the turbines are running at 1400 degrees.

Q: Would you go back and look at the actual values for turbine temps?

A: If I were concerned, yes.

Q: If this delta is high, you know that the turbine temps are high?

A: You'd suspect it. High is good, means a good seal.

Q: If this were too low, it could mean either a bad turbine temp, or a bad seal?

A: Typically would mean a bad seal. Turbines run pretty consistent. Won't run that cool. Maybe will be cool in the bucket, but not at 104%. If they are, you're running off MR or something.

{skip #14, primary seal plot, saw in instr. package}

{#15}

This is Pump end drain temp. If you had significant leakage, this would be cold (lox leaks in). This would normally be about

375-400, maybe warmer.

If it went below 300, I'd be concerned that there was a decent sized leak.

Q: Are you concerned with the slopes of the curves, if it's coming up too slow?

A: I don't really look at that. More of a level check for leaks.

{1 or calc 14}

This is start. MCC pc.

{end of side A}

{David Foust 3, side B}

This is the ICD (interface control document) spec that says MCC pc has to run within those boundaries. We were right outside the ICD on this one, which is a serious problem.

Q: If it ran outside the ICD, would the controller automatically flag it?

A: NO, the controller doesn't recognize the ICD. The ICD is a paper spec.

Q: How do you get that?

A: We have one upstairs.

{#2}

This is MCC pc compared to other calibration test starts. (I would guess). This is an odd start, slow for some reason. Typically, for a slow start, the pc drags out into 1.5-2 sec's then take off because the valves open up and it takes off like a rocket and eventually come in line.

Q: What you're looking for here is what?

A: The pc was a little bit lower, then the engine compensates.

Q: It always comes back in line? If this continued to be the same level off, would you be concerned?

A: Yes, at 3.5 or 4 sec you should be back in line, pretty much should have the engine started, if not you have problems.

Q: Should we expect a pretty close delta for test-to-test variations?

A: Yes Should be about 3006 psi.

Q: But variations "here" are reasonable {1.4 to 4.4 seconeds}?

A: Right. The OPOV command can be set to take care of the undershoot.

Q: These comparison tests for start, are you looking at the controller document to see if you have the same valve schedules?

A: The only schedule that might be weird is the CCV. These are all flight engines so we'd have consistent CCV schedules. The OPOV can vary on the open loop command (set higher or lower depending on the slow starts). Fuel side oscillation also affects slow starts. If you have fuel side oscillation, you typically have a stronger start. We've had contradictions to that rule.

Q: Could a slow start lead you to check to see if you had a fuel side oscillation?

A: If we had a slow start, that would be one of the first things we'd check.

A3-215

Q: Is hotter better?

A: Stronger, hotter, better. It's all saying the same thing. A hot start would be a turbine temp overshooting. On a slow start, the LOX turbine temps will drag way out "here". It's very dramatic, we can't miss it.

Q: So you want a fuel side osc?

A: Yes, it makes starts better.

{ensuing long discussion on slow starts and fuel side osc's that's not transcribed}

Q: Does a slow start affect component wear or life?

A: Only if you have a significant overshoot. There is a redline on start overshoot.

{Another long discussion}

Q: Do you use words like hot or cold at data reviews?

A: No, we use good or strong.

Q: Is a fast or slow start reflected in your prime times?

A: Yes. OPB prime is reflected in the time your lox turbine temps turn around, so it's definitely reflected.

{still trying to figure out what's good or bad}

Flights tend to start slower than ground tests. There's nothing wrong with that. The controller will abort an extremely slow start.

Q: What causes it to be slow?

A: Fuel side osc, temp of propellants. Colder lox tends to start a little stranger or faster than hot lox. These are generalizations. I only remember one test where the start was so slow that we thought we had a problem.

Q: This needs to be noted?

A: Definitely have to mention at a data review.

Q: How much is too much and is this spreading "here" of some significance?

A: Have to look at your valve positions. The valve may have been a little more closed here. There is a lot of variation on the start.

Q: Do you try to track down the variations?

A: We see them so often that probably not.

Probably different engines, but if this was 3 consecutive tests of same engine, we might flag it.

Q: As far as analyzing mainstage, whether it starts fast or slow doesn't have and affect?

A: Right.

{break in tape, seem to have shut off for a small amount of time}

It's indicated here on the pc on A20537, then you really see it in the lox turbine temps and you start looking for fuel side oscillations.

Q: It's indicated here because of this delta?

A: Yes.

Q: How much is too much?

A: That's about what I would expect (about 100 psi, 100-150).

Q: Is "this" combustion phenomena?

A: This is probably the prime time on this guy is a little slower.

Q: Do the models guys model the combustion?

A: Not transiently, but combustion efficiency. PBM does not do transients.

Q: Would the delta start to decrease "here" or up "here"?

A: I would expect that at 3.5 - 4 sec it should be out of there. This is pc, you'd expect pc to be more stable than the turbine temps.

Q: You'd look for a delta of less than what?

A: You have to consider that you're on a much steeper slope here, you can expect a bigger variation on a steeper slope than on a flatter slope. That doesn't concern me. This delta up here is about the same as here.

Q: Up here it's kind of level.

A: Yes, within 8-10 psi.

{#3, FPOV command}

Here's a reaction to the slow start. You can see the valve pos opened up more to speed that slow start. In the 1.5 to 2 sec range you're dragging. By 2.5 the valve kicks open and you see it start to shut down at about 2.8 seconds because you're starting to overtake the other ones.

Q: Down here on valving you'd expect it to be pretty consistent?

A: FPOV ought to be real consistent. That's about as big a variation as I'd want to see and it's probably about .5 to 1 %. At around 4 sec you really can't compare. That's where the actual efficiencies and health of the pump come into affect, more than the actual start sequences.

Q: If you saw a huge variation you'd think it was a sensor?

A: Right. If it was something really large, then I would.

{#4}

Same thing on the OPOV. This is the open loop command where you can increase the levels early in the start which gives you more lox flow and a stronger start.

Q: Would that be in the test objectives or prestart?

A: Yes. I wouldn't be surprised if for this test they increased the OPOV command .5 to 1 % for the next test.

Q: This is commanded so it's not actual measurements?

A: Right.

{#5}

A3-217

This is the OPOV command limit. You're looking to see if it get's much less than 3% between the commanded pos and the limit than I'd be concerned. Not really a problem but a concern.

Q: Do you look at the actuator pos?

A: You could, but the command is what you really look at.

Q: The command limit is set in prestart?

A: It is actually calculated during the start for mainstage. It varies with power level.

{#6 through 10}

This is the fuel side osc. {trend in HPFP ds pr} What you're looking for is this little osc. or bump and this is the slow starting test where there's very little osc.

Q: How much does it have to "wobble" for you to call it a fuel side oscillation?

A: "This" is essentially no fuel side osc.

Q: There is always an osc. but it's a question of to which degree?

A: Right.

Q: Do you look at the temp then?

A: Temp doesn't always show it. We typically get a large pressure osc and don't really see much temperature. It's not always consistent, usually it is, but not always.

Q: If I saw this happening a little more, would I suspect inst?

A: I wouldn't suspect instr until I got into mainstage because this is such a transient.

Q: I would be inclined to flag this as normal since there's really no diagnostics to be done. Is there any engine anomaly that would show up in this graph?

A: You might see a really erratic signal or a definite shift. That might be a sensor scaling problem.

Q: A shift at the start?

A: Or anywhere during the mainstage.

Q: And you would be comparing to a test with a similar osc. profile?

A: You'd probably be comparing to whatever osc's you're comparison tests happen to have. Fuel side osc. is not one of the things I consider when I pick a comparison test.

Q: These look like pretty big level changes, how big would you flag?

{#6} {Careful for plots 6-11, I had a hard time figuring out what was being pointed to, will probably need a closer look}

A: This is FP disch pres. it will depend on how fast the FPB primes. Here, this one didn't have much of a fuel side osc. It will also depend on the temp and pressure of the fuel coming in.

Q: You consider all of this to decide what is a reasonable level change?

A: Yes. This change out here doesn't bother me.

Q: When you say "down here", do you mean below .8 sec?

A: Yes, 0 to .8 sec.

Q: By mainstage you'd expect this parameter to be pretty close?

A: To where it's going to run in mainstage. FP disch pressure will vary if I change a fuel pump.

Q: Is the main reason why you're plotting these starts against one another because you want to see if the engine is starting the same as it did the last time?

A: In this case, no. Usually that is the case, but we don't have a previous start of this engine. We're just looking to see if we had fuel side osc.

We sometimes look at the HPOP disch temp. It will see a spike up.

{#7, #8}

The time frame here is about 1.1 sec, you can look at the fuel turbine temps and see whether it has fuel side osc or not. Obviously this is a tighter scale, this is A channel. This is a pretty typical start.

{#9}

Q: What do you want to flag here?

A: You might pick out a slow start, or a significant delta right at engine start which would indicate that the sensor was reading bad at prestart. Same thing on the B channel. You'll sometimes see different trends between A and B, that's not unusual. They are at different circumferential locations on the duct so B sometimes will run hotter than A.

They will pretty much oscillate in the same way but, for example, on a slow start (flips to lox side) this one actually gets lower.

Q: But you're not plotting channel to channel?

A: No, but we look at a and b on different plots.

Q: Is it fair to say that if I have a dual bridge, and one channel is off from another, that maybe it's an instr. problem, and when they are different sensors I can expect a little more variation?

A: Yes.

Q: But you'd still compare them?

A: Yes.

Q: And what is a reasonable delta from A to B?

A: In start? {yes} In mainstage, I've seen as much as 150 degree difference in fuel turbine temps. In start, I don't recall. Don't look at it all that hard during the start as far as delta. I would say 50-100 is normal.

Q: Do you have to bring this up?

A: We flag it, if they question it, we try to explain. If you see a huge turbine temp A to B delta, we may want to go back and compare it to the last 5 times the pump or engine has run and see if you can come up with any correlations. I've seen as much as 100 degrees on the lox. Lox tends to run, A to B, alot closer together.

{#10}

A3-219

This is B channel, same story.

FP disch pressure. Fairly typical. This variation is the slow start.

Q: That same variation "here" would be a problem?

A: You could see as much as 50 psi difference for test to test with the same pump and that would be normal.

Q: I would flag that, and then say "Oh yeah, I had a slow start"?

A: Right.

Q: How much is to much?

A: It's not unusual to have on a slow start as high as 800-100. Normal is about 300. Something at 800-1000 indicated a problem, in this case a slow start.

Q: And what about between 300 and 800?

A: Then flag it and point out that you see a difference, but not anywhere else.

Fuel pump speed is a reflection of fuel pump discharge pressure. It's a one-to-one relationship. Going from 0 to 35000 rpm.

{#10 is same thing, channel B}

{#11}

HPOT disch pressure. This will reflect it alot more. You'll see a slow start more on the lox than on the fuel side.

Q: Would you have flagged this in the other test?

A: Probably not. Scale of 300 psi.

Q: Because this is the ox side and the ox side reacts more? If this was on the fuel side would you flag it?

{#14}

A: I might. This is your LPFP speed. Same reaction. The LPFP is driven from the coolant flow coming from the MCC which is a function of the HPFP disch pressure.

Q: Do you expect them to be pretty tight?

A: Fairly tight. They are about 15000 rpm and fairly consistent.

Q: Tight is 100 off?

A: Yeah, you probably have 3 different pumps.

{Eric enters and calls us all knot-heads}

{#15}

This is LPFP disch pressure or HPFP inlet pres, you'll see it sometimes on flight with a different name.

Q: This is not flagged as erratic {after 3 seconds}?

A: No, small time frame. If you looked at 0 to 20 sec's you wouldn't see it.

Q: What about this spike {at 1 sec}?

A: That's a typical spike that we see. You're slower start is flatted out. The spike is normal. It may be when you're priming

the PB and get a big surge in FP disch pressure. That will send the same pressure source up here.

Q: Fuel primes first, then MCC, then lox?

A: Think so.

Q: What do I want my delta's below?

A: This is average, so I would say about 10 psi during the start.

Q: If it went in a different direction that would be a problem?

A: It would definitely stand out.

{#16}

This is the LPOP speed, a reflection of HPOP disch pressure. The coolant flow coming through drives the turbine here, coming off main impeller, tapping off, driving turbine on LPOP.

Q: What's normal variation here?

{End of tape}

{David Foust - Tape 4}

This is just a reflection of the HPOP discharge pressure; because, the discharge pressure is what feeds into your turbine. So, pretty much its going to reflect what the HPOP discharge pressure shows.

Q: What are normal deltas?

A: I would say probably on this, on the starter range I would say probably 150-200 psi and then this is obviously that slow start as we were discussing earlier.

Q:Q:O.K., So 150-250 psi is a normal delta?

A: Yes, its probably within the normal range.

Q: During the start transient?

A: Yes, during the start transient.

Q: {question}

That's going to depend on the pump, the big pump, the little pump, the whole engine system. I mean, its going to depend on how the main stage is totally dependent on how the engine ...

Well we'll get to that in mainstage when we hit it.

Low pop speed is a facility measurement that is notorious for being very erratic.

Q: What do you mean?

A: I mean huge spikes, it's a very noisy parameter.

Q: Will we see an example later on?

A: If there's an instrumentation package in here it will probably be in there.

Q: O.K., So the variance is really large.

A: A:And that's just a bad parameter ...

Q: It's a bad PID?

A: No, its a good PID, I'm just saying its just bad notorious being noisy.

A3-221

Q: Do you know what the number is?

A: I should probably bring my data review folder, I really don't know what the PID is off the top of my head.

{#17}

O.K., This is pretty much ??? with low pop speed, this is your high pop inlet pressure or low pop discharge pressure. Same sort of thing here probably you're talking maybe 10-20 psi probably normal on the range, then this is your slower start here. There is usually not too much to say about that.

{#18}

This is your pogo precharge pressure. That's where the helium comes on during start {2.4 seconds}. This is where the oxygen starts again {4.4 seconds}. Pretty much, it's like a valve that's controlled by spring and once the pressure gets high it forces that spring the other way and then the oxygen takes over.

Q: Is this normal? What about the delta here anything in particular?

A: This is going to vary a whole lot, this parameter is not one that is consistent and you can tell that here in the mainstage portion. It depends a lot on the repress flows, it depends on the high-pop discharge pressure, it depends on a lot of different things, but, we're really not all that consistent. This is not one of those parameters that's going to lie around one on top of another, your going to get a pretty good variation of pressures in mainstage. I'd say this is a typical variation for starts, that's probably about 50 psi.

Q: Anything else?

A: No, not really.

Q: Part of a slow start?

A: Yes, definitely.

Q: What about this ? ... here?

A: O.K., that's pretty much like an overshoot. You see how your dragging pretty slow, then all of a sudden the valves open and it kicks in and it takes over and that's how it will slow back down. This pressure and the temperature on the heat exchanger tend to lag a little bit cause they're further down-stream.

Q: What do you mean...?

{#19}

A: This is the interface pressure. Here is your heat exchanger, you're coming in here and here is your anti-flood valve. You come in here and go through your heat exchanger core, your coming out. Now the heat exchanger discharge pressure is like in this vicinity here, but, the interface pressure is where it interfaces with the vehicle or testing and whatever and its further downstream, so it takes longer to get there so therefore, the temperatures and the pressures tend to lag and the engine parameters a little bit.

Q: By how much?

A: The heat exchanger interface temp. for example that we look at, the high-pop turbine temps. come up fairly fast and it takes probably an additional 50 or 60 seconds for the heat exchanger in the first temp. to level out?

Q: I thought everything was considered to happen almost instantaneously.

A: You can start to see it to rise. If you plotted up the turbine temps, then the heat exchanger interface temp. on one ..., it would take quite a while for that heat exchanger to catch up to the ...

Q: So, in that case, if you saw something really high here you wouldn't look at three seconds somewhere else.

A: You might want to look back a little bit in time on that one. Pressure obviously reacts faster than temperatures. But in general, this is further downstream, it is the interface between the engine and the facility ...

Q: Anything bigger than this to expect or would that be normal?

A: That's probably normal and that's probably ...

Q: Anything larger than that, you need worry about?

A: Yes.

O.K., I can probably do this and then we can eye-ball it. If you think that is normal.

Yes, that is normal.

{#19}

This overshoot here, that's due to the slower start, you're coming in here your valves are opening to catch up and then all of a sudden you've gone to far and you've turned them back down.

Q: In any case no matter how high this overshoot was you'd never flag it?

A: If it got really hot, I mean like if the level here were main stage level {careful with this interpretation, may not be correct} and it went way up here, that would be something different. But, you know just a 100 psi overshoot that's not going to be that bad.

{#20}

Q: What about "this guy?"

A: This is the interface temp.

Q: Is this a normal delta?

A: Yes.

Q: It's wider up here than it is down there is that O.K.?

{Flips to LOX turbine temperature plots}

A: Yes, let's look at the lox turbine temps, that'll give you a feel.

You can see here that the lox turbine temps are there starting to go up, you see the slope change here about 2.5 seconds roughly. And they really don't start kicking in untill about over here and that slope change here. It's kind of hard to tell, the trend is a little bit different but, I think this roughly corresponds to that as far as the initial kick-on. You can see here the turbine temps here have been flat for a good two seconds. You're still in the climbing mode here. You're still running up.

{#20???

This is kind of a hard one because you have not only your turbine temps, you have what's called heat exchanger bypass orifice, which is in there, which bypasses flow around the heat exchanger and then it dumps it back in downstream of the heat exchanger and cools the flow down. It's going to depend on the size of the orifice and the engine. I'd say engine to engine it's

A3-223

about average (about 30 degrees or so).

Q: Does the delta need to stay constant?

A: No. They're going to float around depending on the engine, etc.

{#21}

This is the anti-flood valve pos. That's the spring operated valve in front of the heat exchanger. As soon as the pressure gets up high enough, it'll open that valve up.

Q: Do you ever look at this time with respect to when the pressure got high enough to see if the valve was sticking, etc?

A: If it looks like a valve was sticking, we would. Otherwise it's pretty consistent on when it opens.

Q: Is that important {blip at .3 seconds}?

A: I wouldn't have called it out, it may be some kind of pressure blip.

Q: Does this pop if it's higher than 120?

A: This is percent open, not pressure.

Q: These variations are normal?

A: Yes. Some valves read greater than 100%, that's just the RVDT. The ccv and anti-flood valve often read 101 or negative, the mov and mfv will sometimes read over 100%.

Q: What's a normal delta and is that important?

A: Not really, we don't look much at the anti-flood valve unless there's a sticking valve or something. I'd say about 3% is about normal.

{#22}

This is the intermediate seal purge pressure. The main thing to look at in the engine start time frame is level. The flight min is about 175 at engine start and that's an LCC.

Q: Do you care what goes on here?

A: We really don't look at it much during start. You'll see that these blips are fairly consistent. This dip and surge is when the pogo precharge kicks up and goes off (dip at 2.4 sec, surge at 4.4 sec). It should line up exactly.

Q: How about this upward trending?

A: Remember mainstage? It's just starting to come up.

Q: If I saw a downward shift, would I worry?

A: Yes. I've never seen that because you're powering up, everything should be going up.

{#23}

These are bit toggles. That's normal, just the accuracy. These are the shutdown purge pressures. Basically, after shutdown, you blow in a helium purge to blow out any residual gasses that are in the combustion chamber so you don't keep burning after shutdown. It blows them out the hot gas manifold and out the nozzle. The requirement is they have to be less than 50 psi during mainstage to guarantee that they're off. So, they're about 120 on FPB and 450 on OPB during prestart and post-shutdown.

Q: Do you give a block I parameter twice the leeway for in

determining it's meaningful range because it has twice the bit toggle?

A: No, the controller has the same limits. I look at the average, and the bit toggle doesn't make much of a difference.

{#24}

This is fuel system purge pressure. It has to be <50 psi during mainstage. That comes on for 3 min. every hour during prestart. We normally don't see any drifting on these kinds of things.

{End of Start package}

{shutdown package}

{#1}

Let's go into shutdown. This is the pc. It's a 100% shutdown. You will see a different trend if you shut down from 90% or 104%, etc. It tends to change the trace slightly, but these are pretty consistent. This is a typical shutdown.

Q: Are these the same comparison tests that you used for start?

A: I think so.

Q: Could you use a different test for shutdown.

A: You could.

Q: Do you?

A: No. I wouldn't if I know I was going to shutdown from a different power level. Otherwise, I like to keep things consistent.

This is a pretty consistent MCC pc for shutdown.

Q: Normal delta?

A: This is pretty average. They should fall on top of one another in "this range here".

{#2}

These are fuel turbine disch. temps. Nothing significant. We look for leaks during shutdown. Make sure all the turbine temps warm up. If you're leaking past the turbine seals, the turbine temps would stay cold. We look out to 300 sec's after shutdown. This is an average delta. I've seen some bigger. It get's a little tighter "here". Another thing to keep in mind is that sometimes, depending on the engine, MR, etc, the fuel turbine temps may be really hot. This will affect how long it takes to get down. Some thing if the turbines are running cold - it will come down a little faster.

{#3}

This is a typical delta on the B channel also. Here's a good example of the variations you can expect.

Q: No significance about the pattern?

A: Not really. It will vary from channel to channel. You tend to get a bump in channel A, B is smoother. That's fairly uniform.

Q: As big of a delta on this one as on the last?

A: A little smaller on the lox pump. Fuel pump tends to have a bigger delta.

A3-225

Q: How much?

A: This is about average.

Q: It's really high here, do you disregard that?

A: The delta kind of goes out as you shut down. That's not unusual, depending on the engines and pumps you're comparing, to have a variation in starting point. This is typical, considering that the turbine temps are starting out at a fairly decent delta. Slightly different trend on B channel but same story.

Q: Other than sensor stuff, is there much in shutdown that can help explain what happened during mainstage?

A: There are some things we look for strictly at shutdown. Rotor grab is one example, you'll see a significant drop off in speed - it will all of a sudden shut down and be flat. Sometimes you'll see it in the discharge press's and speeds, depending on the pumps. You look for cavitation during shutdown. Especially during flight, you'll look for lox pump cavitation. As far as the transient shutdown, you might see some parameters recover from being frozen. It might be frozen and not respond to the 3g throttle, then it will kick back in to normal during shutdown. Mostly you'll look for bad sensors or erratic sensors. If it was erratic in mainstage, is it still bad during shutdown. Looking to see that the speeds and temps come down normal. Also the amount of time it takes to spin down.

{#6}

Q: "This" is no problem? It cutting off early there? and this one going out?

A: That looks like extra energy in the fuel PB. That's a little abnormal to go out as far as it is. "This" is more typical, on these two. Fine preburner pc and that'll tell you if it's abnormal or not. You really don't see it in the pc.

Q: If something starts slow, does it stop slow?

A: No. That's more prime time and how the fuel is getting there. This is strictly a matter of how much momentum is in the pumps. That's a little long, but there's not anything we could do about it. They do torque checks after the engine is run anyway. That's something to note however.

Q: When does it usually cut off?

A: Usually between 15-18 seconds is typical. Anything over 18-20 is starting to get long. They aren't going to do much about, it's just unusual.

Q: Is this the only time when pumps spin down?

A: Yes. Throttling, but this is the only major speed change.

Q: "This" kind of a delta is normal?

A: That's a little big. It looks like it's got more energy in here that I don't understand. I didn't see it in the FPB pc.

Q: Looks like it got a boost of energy "there".

A: Yes. It looks like something bumped it at about 6 seconds.

Q: What's "that"?

A: This is not too abnormal to have a little variation "here" during spin down. A lot of that may be in the pc (we'll take a closer look at pc). That's about normal. This is a little abnormal - it hanging out like that. We'll take a look at the turbine temps as we go on too.

{#7}

This is HPOP disch press, you don't have HPOP speed on flight. You have to get it from the dynamic data and they only get it for mainstage. You never have HPOP speed during shutdown. This is a typical shutdown.

Q: This gap is typical?

A: Yes. That's about average.

{#8}

This is FP disch. pressure. This is more typical on the top two, this one is probably (the triangles) the slow one.

Q: Is that corresponding with that?

A: This is about 2.5 seconds.

Q: So, that's probably normal?

A: Yes.

Q: The cut-off of those speeds, is this the normal range to expect that?

A: Yes, that's about the normal range and then this guy is probably, part this is due to the fuel pump that's kind of spinning down slower. Its probably pumping a little more flow through the coolant leg, which is powering your low pressure fuel pump. Typically, when things in your big pumps are going to run then your little pumps kind of reflect it, because they are being powered by the big pumps.

Q: Normal the pump and everything?

A: Yes.

Q: Normal gap up here?

A: Yes, you're starting at a higher speed.

Q: Would I expect any bigger delta?

A: That's about average.

{#10}

O.K., this is the fuel pump inlet pressure or the low pressure fuel pump discharge pressure and that's pretty typical. We saw it fallen off about 6 seconds, 7 seconds and you really don't ...

I really don't know which one that is.

That's kind of hard to see, but you can see here it doesn't vary a lot and you're already down here below 50 psi, so your getting down there where its spinning down. That's pretty average as far as the shutdown.

Q: Right, that's the normal delta, what would be too big?

A: That's probably that one spot we saw before. That's probably about 2.5 and this is a little earlier than that. This is about average.

Q: From here to here?

A: Yes, that's not unusual.

Q: Normally it spikes up?

A3-227

A: Yes.

Q: Why?

A: It's got to be a change in your delta p across from your turbine on the low pressure fuel pump. I am trying to think of what would drive that. You're shutting down your big pump, so, your main fuel valve is closing. Its probably due to the fact your high pressure pump is going down fast and therefore its ... Lets see downstream your coming in here and then your dumping down eventually into your turbine discharge. That's a good question. I'm trying to think of what would drive that. This valve's already shut now.

Q: So nothing is going past here anymore?

A: Right.

Q: So, its whatever is left.

A: It's your delta between here and here that's driving your fuel pump, your low pressure fuel pump, and your driving pressure is actually dropped down, but your resistance is probably still in here. So it is a greater delta p, that's probably what's driving it just for a second.

Q: So for ... you don't have as much going through here get warmer and get hotter because, there is not as much film. You know because you only have that residual that was left in the line so that makes it hotter or no?

A: Yes, it would get a little warmer, but your combustion has pretty much stopped. I'll have to look into that. I don't know what draws it up. Low top speed is a pretty typical shutdown.

Q: So its best to shutdown between 12.5 and 15 or even tighter?

A: That's about right 12.5 to 15 seconds. Its going to follow your high-pop discharge pressure because that's the driving pressure in your low-pop. This is just the discharge pressure from the low-pop. Your going to see a lot of variation here. This in not untypical for variation. A lot of valves are closing and you see a lot of water hammer effect in your valves. Its just where a valve closes and you get a pressure wave back up through the flow. That's pretty typical and its going to vary a little from engine to engine so, that's not unusual and your going to see variations in the spikes. This is pretty typical for a low pump shutdown.

Q: So this would be normal?

A: Yes, that would be normal.

Q: Alright spikes should occur in these places.

A: Yes, they should pretty much occur in same places, but the level of them can vary. That's not unusual for them to vary that much. O.K., that's the bad fuel ... that was bad, its gone below 0 psi, so, that's the one we talked about from the data review package.

Q: What's normal delta then? It seems pretty tight.

A: That's about right. You could see you're starting from different levels. This one was one that was bad, but you can see you can vary a little bit up here at the starting point. There's a bad point measurement.

{#14}

That's the OPB pc that was bad.

Q: What's the normal delta? This is really tight. Is it a little

bigger than that?

A: Probably a little bigger than that. That is a little bit tight. That's probably 75 psi, that's probably not too bad during the shutdown.

{#15}

This is your fuel PB pressure for shutdown purge. It kicks on at almost two seconds and then you come down here and this is another purge. It is all helium.

Q: Is this due to the helium?

A: Yes, due to the helium being used in other places and that's pretty typical. That's about an average shutdown with three different tests and three different pumps. This is the lox ... purge.

Q: ?

A: That is 16., that's where the pogo precharge shuts off and causes that little ... We'll take a look at the pogo ... All that stuff is driven by helium, so, anytime you turn off one purge and turn another purge, your going to see a little blip in the other parameters.

Q: ?

A: That's just the valve opening up. I think it's a spring loaded valve and there are a lot of lines that you have to fill in those valves, so that's pretty typical. You can see here that's pretty consistent, with 23 different engines and three different pumps.

Q: What's normal?

A: That's pretty typical right there.

Q: Any bigger?

A: Maybe slightly, but, that's about right. Here's your pogo precharge.

Q: How about these things normal?

A: Yes, those are normal. Its helium. Everything is pretty much helium.

Q: Delta normal?

A: Yes, you see a little variation in the ... that shutdown, just like you do here in mainstage.

Q: But, it's supposed to be much tighter here, or if I had something that big here it's fine?

A: Yes, this is helium through here.

Q: So, it's real tight?

A: Yes. This is the same kind of ... during shutdown.

{#18}

This is your spring-loaded anti-flood valve. Its going to close whenever the pressure drops off enough for the valves to close and you can see that's about the same that it was during ... Little over 100 percent.

Q: So, that was something like two-percent.

A: Under three-Percent. O.K., this is your intermittent seal

A3-229

purge pressure That's what shuts off. That's your 16.... You can see here they vary during mainstage, so they kind of hold that ... and they all shut off at the same time and kind of all go down to about the same level; Which is psi, I think.

{#19}

Q: Is that delta normal?

A: Yes, that's fine, its just the operating level.

Q: So, what you need to do is check the difference there?

A: Yes, and if you have a huge difference in mainstage, you're probably going to have a huge difference in shutdown and probably during the start as well. They are going to be pretty consistent. It depends how tight the seal is. A tighter seal is going to have a newer pump, where a tighter seal is going to higher pressure than an older pump with a little bit more worn seal.

{#20}

Q: Is this normal.

A: Yes. O.K. this is whole bunch of different valves that we show and they are kind of confusing. When you're starting your engines, or before you start your engines, your bleed valves are open. As soon as you go to engine start your bleed valves close. Actually, its a little before engine starts that the bleed valves close. So, after shutdown, they open back up. They are your x's and your triangles. You can see they are close to your and then they open up about 16 seconds.

Q: What do I want to check, the delta's need to be the same?

A: That's just two different tasks. This is going to be the same kind of thing with your anti-flood valve. You probably have two or three percent delta up here would be O.K. The thing your really keying off on this thing here is to make sure the proper valves close and the other valves open up.

Q: Do you ever check these ... here and go back and put the mainstage ...?

A: We really don't flag it out unless its pretty significant. Because, its just that how the ... reads or it depends on rotary valve or a linear top valve.

Q: Do they tend to shift?

A: Not usually. You will see sometimes on a mainstage if you look at your main fuel valve and your lox valve. You know right as an engine starts, they are going to open up and then they will be at 100 percent. During mainstage it will go up from 100-102 for no apparent reason it just go's up and that may be some kind of thermal transit; but, the main indications your looking for is that the valves are opening and closing when they should be at 16 seconds.

Q: Why did delta, during the main stage that I said was normal, because it was in that three percent you said and then I looked and maybe it closed way negative. I could look back at mainstage and I was back by seven percent and I didn't know it. Would that be reasonable.

A: Yes, that would be reasonable. If the delta changed significantly from mainstage to shutdown or from start to mainstage or whatever.

Q: If this went negative and it was a wee bit high during the mainstage than we thought what you said was as high kind of takes ...

{#21}

A: O.K. that is your emergency shutdown pressure. What that is, is pressure that is given to the controller in case there is a problem and they have to shut down pneumatically. Its just the pressure that allows the pneumatics to override the hydraulics.

Q: So anything important here?

A: It has to be less than 50 during mainstage and comes on in about 14 seconds. That is a facility, its actually a interface pressure measurement, but, that's pretty typical in an emergency shutdown.

Q: How come ...?

A: That may be that the helium is getting used somewhere else and it kicks back up. I'll have to take a flip back through. It goes up at 15 seconds. There is probably another purge or something that's kicking off right at that time. See it kicks up here in the intermediate seal.

Q: So, that's normal?

A: Yes, that's normal, it happens all the time. Main thing you look for in a manifold?? discharge temp is your looking for leaks past the manifold. This is prestart. I think the limit is 320 degrees that indicates a leak.

Q: It has to be greater than 320 degrees?

Q: Is there anything marked in prestart package as far as limits?

A: Looking in here you're about 460 to 0 degrees fahrenheit, so your running about 9 degrees 80 degrees or so. This is where that fuel system purge comes on, its always going to warm you up slightly. This is a fuel system purge here, right there, then you have a fuel system purge, it comes on continually. Its three minutes every hour, but, when you get into purge 4 it comes on continually and stays on that's why it comes up and warms up. Basically your looking for the minimum level. If you got down to the 300 range and you could be talking maybe about a slight leak past your main flood valve, you would be considered about that.

This is your anti-flood valve basically your looking to see to make sure both the T1 and the T2, these are temperatures, its a leak protection as well.

Q: Is delta normal here?

A: Yes, its about average for delta.

Q: Could I expect this kind of delta, in this area too.

A: Yes. Same thing you're looking for anything to be really cold. You know you're lox is around 100.

{looking through a package}

Q: Is it all in there, so, we would have to go through there and they would have all those limits for us?

A: Most of them should be in here.

Then that is fine, because, we could do that.

Here it is, anything greater than 380 degrees range. Main flood valve has to be greater than 260 that was the pop before.

Q: Are you looking for any kind of spikes?

A: Not really in this stuff. You're looking for leaks, that's strictly what your looking for.

A3-231

Q: So, you're just looking for levels in major ?. If this was really bad, then you would think it was a sensor?

A: Well it depends on what the measurements in or what the spike is in. Typically spikes are going to be instrumentation. You don't see a whole bunch of spikes that are real and there are things that happen so fast that the engine can't react. They are so small with the engine that reacts.

Q: They wouldn't tell you the company.

A: Mostly in the ____? but you could have them in the engineer area as well.

Q: Normal, wouldn't flag?

A: No, that's not that bad. This is another leak detection. What it is

that there is actually a little bubble the lox go all the way to the O.P.

When you're pressurized, you come in here you have lox all the way in here you have lox all the way in here, you have lox all the way down to the main oxidizer valve. You also have it going through all the way preburner and you have it all the way to the OPOVCS schematic loss, so, its not really accurate, and all the way to fpov and those valves are closed. Actually, the way the OPOV is oriented, is kind of pulling it up and there is a little room for a bubble, and there actually is a bubble in there. If its leaking that bubble will bleed through and it will actually be cold and then these temp sensors will drop down. These are outside like skin tents.

Q: Will the bubble pop?

A: No, if the bubble went through and it leaked. You'd have liquid lox you'd be dropping down in the hundreds.

Q: Any kind of normal?

A: That's about average.

Q: No more?

A: It wouldn't concern me even if it got as big as two or three degrees. That really wouldn't bother me that much. It just depends on why we want to overlook at that except prestart basically, strictly for late detection. O.K., these are your fuel turbine temps and lox turbine temps prestart. This is the result of a purge coming on. A lox dome type purge that comes on and basically warms it up a little bit, warmed helium. Your main concern is the levels, you want them to be around ambient. 530 is what? 70 degrees or something like that, 70 degrees fahrenheit, so, that's about ambient. That's fairly what your looking for, that should track fairly close to one another, cause there is no flow going through them, so, the turbine temp should be fairly close.

Q: ?

A: Not necessarily.

Q: This is (?)one's lox from one's fuel.

A: Right.

Q: So, how big.

A: I would say, you are talking five degrees here, that's pretty tight. I'd say if they were much warmer like five degrees off of them I might start looking a little harder.

Q: So, If you were closer to watch one, There wasn't say a big difference..?

A: These might fall right on top of one another, that would be your concern. They are all in the hot gas manifold. So, they probably wouldn't be a whole lot different. Typically, your going to see a delta, but, it wouldn't concern me a lot if I saw all the lox ... laying around on top of the fuel.

Q: But, they also have to have the same trend.

A: You are going to see that trend, because that's a heated purge that comes along, If you don't see this rise in here, then something is kind of unusual.

Q: This is all normal?

A: Yes. O.K., this is your loxdome temp, same kind of deal as far as the ... here, its just a heated cartridge that comes into the logstone. The logstone is just up in here, its where your lox comes down into your, they call it the hotdog. Its just the temperature up in this area, kind of a leak check also, because your range loc valve is closed. If anything would leak and pass it, the temperature would drop like a rock. So, its kind of another way to check a leak.

Q: If you saw a spike, ...?

A: If you saw a spike prestart, because there is really nothing going on other than a purge blowing in. If you did see a spike, I would probably flag it and go look at the helium supply pressure, to see if for some reason it was spike in the pressure that would drive this.

Q: The level checks are found in that one package?

A: Yes, they should be in there. They call it a leak detect package and that's what you are looking for.

Q: What size frequency ... are actual?

A: That's the recording rate of the data. They changed the recording rate, you can see here it picks up out here. This is probably like one sample a second, it may be one every ten seconds, and then over here its probably higher frequency; because, there is probably something going on. Its higher out here toward engine start. They do that all the time in prestarts. O.K., these are facility measurements, do we need to go into those?

Yes. This is a bad sensor, because, it is so erotic.

These are gloclogs, there thermacouple. What you try to do here is pick up, they are for fire detection.

Q: Yes. He explained that there is supposed to be allowed a large variation?

A: Yes.

Q: How large? Can you give me a handle?

A: That is pretty erotic.

Q: That's only that particular sensor too?

A: Yes, I would say those things 10-15 degrees, really wouldn't bother me as far as the variation between positions. This is northstate, northsouth, east and west. If you pick a fireup its going to go through the roof. Unless it drops down with wind you don't worry about it. These are amnia-power attempts. The difference is the position in the stand and where they are, these are the glocarts are there in case you have a hydrogen leak. This is like a little lighter that will ignite that leak and go ahead and set it off, so, it won't build up and cause a large explosion. It will just go ahead and ignite it and then it will let it burn. Then, the thermal couple will pick it up at temperature and the facility will cut you off. This is more for a fire that's up

A3-233

around the power head, so, this is actually up around the engine, that's more of a facility type. This more of an engine run the power head. Its the same thing, its temperature sensors, probably the same range as you see here its a pretty tight scale. But, this whole spike here, looks pretty big, but, your only going from 530-520, so your only about 15 degrees.

Q: So, I still say 15 degrees, I wouldn't flag anything?

A: No, Now, these are the north lower, east lower, south lower, west lower, these are the upper. So, they have an upper range, and lower range, to pick up fires up on the engine or down on the engine.

Q: Anything interesting that this is spiking at the same time that this is spiking?

A: I can't see the time scale, so, I don't know what could possibly be going on at that time, but, it could be that they are pressurizing the system. This is another leak detector that we use. This is the Venturi inlet temperature on the fuel pressurization line. We come through here and you have fuel through here all the way to your big pump and you stop here. What they are checking is that you are not leaking past the seals that are in between the fuel turbine and fuel pump on the low pressure pump. If you leak here, you come here, here's your repress line. It goes off to a burned stock on the ground and it goes to the external tie. If you were leaking past the seals and stuff, you would probably cooling this down quite a bit, because its liquid hydrogen. So, what they are doing is looking for anything below ambient or significant here and its just a venturi inlet temperature that's probably downstream a little bit.

Q: So, its a level thing?

A: Yes, its strictly leaks. In the 550 range.

Q: Nothing important about the ...?

A: Right. O.K. this is your eminent seal pressure.

Q: Isn't this like a bad data point?

A: Yes. There is when you go from purge one to purge three, there is like a quick little blip of data or purge, actually nitrogen back here. Anytime you go into purge three, they have like a ten second shot of flow going into your intermediate seal, as a protection. Because, the reason that came about is not only a couple of years ago, we were in purge four and we had to go back to purge three. Well, in purge four we were using helium bottles on the orbiter and purge three we were using nitre-ground nitrogen. Well, when we were in purge four, they shut down the ground nitrogen pressure with a rubber back from four to three. It took them several seconds for them to bring back the nitrogen, so, your sitting there with no purge in your intermidium ... So, what they decided to do in the controller was anytime you go into purge three, whether its from two, three, or from four to three, leave the helium on for an additional ten seconds, that will allow you time for you to get your hydrogen back up. That maybe what this is, it may not. I don't know.

Q: ?

A: Should be a ...

Q: We can get it if we need it. The other ... don't have anything, they are all screwed up too. So, I guess the best thing to do is leave it with this one.

A: You can pull this test and see when the data starts and that will give you a feel for a scale.

Darryl Gaddy on Sensor Validation interviewed by Claudia and Tim.

A: Performance has changed or the instrumentation has gone bad, if you have a redundant measurement like 2 sensors measuring the same pressure you can compare them to two difference measurement is a good way to see if one of them has gone bad. Or if you have another one downstream or upstream you can compare high-pressure lox pump discharge pressure we also have injection pressures, you can compare those two to see which one is bad or see if one is drifting.

Q: Do you have these flow diagrams? (Katherine was trying to get us colored copies...Does it have the instrumentation locations?)..

A: Yes, it has the flight instrumentations, it has the antiflood valve, it has position A and B, and MSID numbers and the pid numbers. Like if we see high-pop discharge pressure sensor here, if we have cad measurement we also have facility measurements, so there is a cad and facility so if you compare the cad and facility measurements together and one of them is going bad it says it the cads part of the measurement or the facility part of it. Or high-pop discharge pressure here you also have a lox injection pressure that is down here, I don't see it here but we do have it, so this pressure and this pressure should be the same and if one pressure is increasing then it should be increasing.

There are a couple difference categories of instrumentation failure that you can have, one where you go completely off range within one measured time that is good, within 20 milliseconds or 40 milliseconds is bad on the next one and that just a thermocouple or a sensor failure, or something breaks in the thing. There is no good way off scale or below scale. The other failure you can have is like a drifting measurement, you can have a measurements that just drift. This is on the last flight we have pid grid on PC the measurement and the fuel preburner chamber pressure, see it comes up here nice and steady, then it just drifts down like this. This is characteristically what we call a thermal drift, it gets longer and longer and the temperature of the engine affects the measurement. This one we have a fix for it, we have offset valves, we have the pressure tap come up, we put a formal isolator on there and we ran a little sense line around in a circle above the isolator.

Q: That wasn't the case here though?

A: This one did not have the isolator on the offset valve so it drifts but the ones that do have the offset mounts are nice and flat and like for this one the fuel preburner chamber pressure, to look and see if it is really drifting, I would look at fuel pump speed, something that this pressure is driving the system to make sure that it is actually drifting.

Q: What kind of pre and post test measurement do you look at? Do you use those too?

A: Right now we don't look at it, or they might have program that looks at it. Looking through the data package, this one is drifting I'll go back and look at prestart post shutdown to make sure it is reasonable or where they are suppose to be. (okay) This one i would expect it to drift below 0 psia. All the cads measurements there are sensor qualification documents where we will get a fid if the sensor goes above a certain value.

Q: The other thing that we need is a translation on those fids, do you have that?

A: Someone that worked over in the controller group gave it to me, its a wordstar/ascii format and I working out and we are going to put it on the SUN station. I'll have it resident on the SUN station as INGRESS database, whenever I get it ?. This is the database, this is block two, and you get a fid and it will post out a fid and this is the measurement in the data stream like pid. No its a failure id. Its word number four, which is pid number 4,

A3-235

data work 5 pid number four, it spits it out as decimal you convert that to octal and you get the fid. That how you get that one and you just come down this listing. Come down to the 015s and come down 041 which is FPOV channel A first failure. This even tells you during the check out phase the response that the engine is going to take, it will say disqualify channel A and inhibit start. This one will do actuator channel disqualification and inhibit start and if your start you will disqualify that actuator.

The controller can tell you whenever we have failures on certain measurements, not all the controller measurements have qualification on them.

Q: Just ones like for redlining control?

A: Right. Another good way to look at comparisons for instrumentation failure if you don't have any hardware changes and your in the same inlet conditions of power level you can compare to the previous tests.

Q: That's a good way to figure out your engine are pretty well tuned right

A: Yes, you have to know the engine performance is the same here too.

Q: A lot of people mention the drift on the fuel preburner, and in a couple other places, is that listed somewhere or written down?

A: Of what measurements are bad? I don't think so, I could tell you, like the fuel preburner it always drifts, I think it's pids 17 or 18 MCC coolant discharge pressure, it drifts a lot, we don't believe it. There is a measurement that comes about here this pid 17 measured here also on the ground test have a ground test over here before we go into the turbine and these two partly agree. This one is pretty steady and this one is the best. We've having a lot of problems lox discharge on certain stands but that not one you expect to drift all the time, that pid 334. No may of them you expect to drift.

Q: And what about ones, like I think pid 30 is often bad?

A: Yeah, that a really bad speed measurement on the low pot, and pid 260 and 261 which is fuel big pump speed, they have a really big band to them, like a couple thousand RPM toggle, where the pid facility ? 764 is really tight.

Q: Okay, so you would use?

A: Yes.

Q: The other thing is you do a lot of comparison between cad and facility, sometimes they are right on top of each other, sometime they're not and in either case it could be acceptable, those are not databased somewhere either?

A: ENGSYST program Martin Marietta has written, have you seen that, it has a p? in the program.

Q: I've never heard of that program, what is it?

(too many talking at same time/can't make it out)

A: This is a really good code. They have it here with the tolerance, down here and you have to go back up here and get your pid number.

Q: To interpret what the pid is...

A: I think if you run the program it puts it on the screen, what the tolerance is and the pids.

Q: You mean not all the redundant ones are included in there?

A: These are the redundant ones.

Q: I'm talking about deltas between A and B.

A: ...right, I think they did put that in here.

Q: Sometimes it bothers them when they're slight off set, the times can be quite a bit off?

A: Turbine is sure to start the temperatures, that can be 250 degrees apart, the fuel flow meter discharge temperature is a 100th degree it would bother me.

Q: What are they based on?

A: I think, our experience, we looked at the data and then they coded it into the program and it spit out errors and then went back and looked at the data to see if it really looked bad. I think, but I didn't write the code, I'm not sure.

Q: More like a trial and error thing?

A: Yeah, I'm not sure what it is based on.

Any questions:

Q: Other examples, like there is the icing up problem where you can get water and MC2s, hot gas injection?

A: Yeah, pressure it ices up a lot.

Q: Is it a cads or facility?

A: Its a cads, its the 24, its also a facility that is 367 and another facility that is 371. I'm impressed with your pid knowledge. (24 doesn't often times does seem to be) often times its hanging on, it doesn't respond, they just don't have the measurement hooked up, they've used it for something else.

Q: Something that David Foust was talking about this morning, that I had never head before, there's a redline parameter for 53 and 54, the coolant liner pressure. He said they ice up, because it's sensor, because its actually in the line.

A: Here is coolant water discharge orifices which are down this way and that is where the hydrogen dumps back into the turbine drain, and you get mixing of the hydrogen with the hot gas coming through there and it icing up the discharge orifices and that drives the pressure up, so its really the sensor reading a real phenomena.

Q: That is what I understood he meant this morning too.

A: It is not a sensor icing up as in the MCC hot gas injection pressure, that is something wrong with the sensor, the coolant liner is actually a coolant liner anomaly.

A: Are you not worried than there won't be enough coolant going through?

A: Yeah, that is the big worry that you'll ice up the orifices and then you won't have any coolant to go through the coolant liner and you'll have a burn through.

Q: Is there lower, I know there is an upper limit the redline is an upper limit, so is there a lower limit that would shut it down too?

A: They usually break up on every icing, they do trip up on every icing up orifice, because it is getting pressure from the fuel system, that is a higher supply of pressure, so every icing pressure is kind of low. And that redline is based on pc.

There are a lot more coolant liner on flight than on ground

A3-237

testing, because everybody likes to look at them. Rocketdyne gave us summary of coolant liner shifts, and I made a table up.

You only have to worry about the icing when it is exposed to the hot gas.

Q: One thing that we have heard about several times is oxidizing preburner chamber pressure that facility did that icing up and that seems to be a sensing problem?

a: We don't have a cad measurement there.

Q: Where else would icing up like that happen?

A: You never have anything icing up with anything unless its exposed hot gas, like your OBB PC, we have a big problem out here because we are heavily instrumental on test bed and hot gas area, like the pressure going in here all these measurements we have to purge them before start, purges the line to get all the water or whatever out and then between power level changes we purge them, that way we don't have the chance to getting a delta P that drives the all the gas up there. We have a lot more on test bed than anywhere else.

Q: A lot more icing problems or purging to prevent?

A: Well we purge to prevent but we have more instrumentation on this region. I think that all the instrumentation we do have, there unless its special instrumentation. I don't ever remember fuel burner pc ever icing. Anything in the hot gas can ice, and the reason it ices, is that we sense it, we don't have the transducer in so we send a sense line through and the power head has a shell in it so we are running hydrogen on the out cut cooler, but we have to run this little sense line through to get the pressure outside, so you have this little line with hydrogen running.

I wanted to show you we had a flight done on instrumentation anomaly. On this test 44, we had a MCC PC bias where we have channel, on MCC PC we actually have four measurements measure the PC and then we have A1, A2, average of this one - channel A average and we average these two for channel B average and we take the two averages and average them for a real PC. On this flight we were there is a 35 psi difference between channel A and channel B difference. One is really higher than the other one, the measurement was way out here we sense it through the acoustic cavity, this is the main combustion chamber over here, we have a little purge system on the shuttle engine where we take fuel, warm fuel, and purge it back this way, to keep this line from icing up, and afterwards they found the green gel around this orifice and apparently it clogged it up, for a while then we just blew it out.

Q: So when you first saw this data, did you know immediately it was an instrumentation problem?

A: Yeah, because we knew we had a PC bias, but we didn't know why and we didn't know which one was right.

Q: How did you know you had a PC bias?

A: We saw this one in real time as it was flying, we get data sent over and we have a channel like MCC PC and we have a plot next to it the deltas between to the channels and usually it lessen by psi and it was reading 35.

Q: But how far apart are the transducers and were you confident it was some physical phenomena that was occurring?

A: They're right on top of each other.

Q: No I mean, physically located.

A: They are 90 degrees apart, it is not a location problem, its a transducer problem, if they're not reading the same thing, there

is a problem. They normally run right on top of each other like here. So we saw we had a difference between the two, so to get the engine to run to the same position, one of these is actual pc, so that the real pc maybe reading or maybe reading here. Maybe running a lower or higher thrust and based on that effect your residuals and your ability to make orbit. JSC has gains for different failures that affect performance and they can tell if your running high or low. I've got this table if you want copies (yes) like if your running at a lower pc your lox pump discharge pressure are going to be lower because your going to have to pump up higher, and they have different stages, like they call it a step one MPC PC high real low, its reading higher over here but really its driving the real pc. Another real good one on how this one recovered is whenever it recovered we were reading down here and this measurement came all the way down here and then it controlled back up together that's a pretty dead give away. Like we changed lox pumps before to the last flight so you couldn't tell the lox pump discharge pressure if your engine was running high or low.

Q: But if you hadn't changed the pump you could have told?

A: Yeah, it was a really small it didn't affect performance its less than a power level, one percent power, 103.2 percent instead of 104 percent power level.

Q: Sort of recap..some of the stuff we may do and see how reasonable you think it sounds. One of the first things is take advantage of the hardware we done, or A and B channels cads versus facility.

A: Or cads versus cads or facility versus cads.

Q: Take differences and compare them to ? in that programs, and with those we understand that they may have to be refined. For sensors, maybe for all sensors, the modeling group looks like some zero shift check and maybe be able to incorporate that into the c codes, check for those kind of problems and then there are quite number of sensors that don't have any hardware with them, we have to use reasoning to related parameters also shift at that time?

A: There is also looking for different signals and characteristics, like the skin temperature where it gets noisy and then goes upscale.

Q: Okay, and the low-pot speed is erratic, for excessive noise, that would be something interesting, sometimes you say it, like except for pid 260 you expect a 1,000 RPM?

A: Or even 2,000.

Q: How, do you just know that or is it written somewhere?-is that in that ensign program?

A: No, its not in there, that how we are trying to write this system module, so many people's brain around here they are starting to walk out the door. No that's not written down, but I think if you asked us for it we could write it down.

Q: Most of the sensors we have tables of accurately for them, most of them are two percent full scale, is that about the right ballpark that you use?

A: Most of them were better than one percent, most of them are about half percent. This is something we knew we were going to do a year ago when we started doing this documentation and on main chamber combustion pressure we were trying to list all those requirements that we have on these measurements. Like prestart between purge control and engine ready it has to be between zero and 37, and if not there is something wrong, but we'll get a fid on that one because it is a controller. We have all these things, and we were going to go through and put all the sensor value and that stuff but we never did.

A3-239

Q: Can we get a copy of that then?

A: Yes, it goes through most of the cad measurements and it tells you where the specification, what requires it and what the value, does explain a lot. A lot of this is so much experience looking at the data.

Q: On a lot of the Martin reports, it says "miscalibrated", that might be something the sensor validation module should screen...

A: You have two flowmeters on the facility and on the engine and the facility and engine mass flow rate should be within 1/4 of a psi.

Q: But it senses volumetric flow right? Do you have the equations to calculate this from t's and p's? And, do you use codes with hydrogen property tables?

A: On the PE (perkin elmer) we have a routine that goes out and has tables on ? Right now I have an equation that may be available in Cosmic. (Gasplus). Rockwell people go thru tables and try to curve fit them to the ranges that we use.

Q: Do these work pretty well?

A: I don't know.

This states that "it" is as good as +/- .003 pounds per cubic foot for densities between 1400 psi (?). We'll either have lookup tables or equations on the SUN. Maybe we'll have it in an INGRESS table for you. If not, we can give you the equations.

You talk about the flowmeters being uncalibrated, we talk about the engine fuel flowmeter...we always assume the facility is right. They know the volume of the tank, and can calculate the flow pretty accurately. Every like 6 months, you have to calibrate the facility flowmeters, so if the engine mass does not equal the facility mass flow rate then we adjust the engine flow rate.

Q: What do you adjust?

A: The software constant, Kf.

Q: That means that you had misjudged the efficiency of the flowmeter?

A: Yes. We usually do a water calibration. We've also seen the flowmeter change from engine to engine. Changing out a calibrated flowmeter can change it because of the new engine.

Q: This is being calculated down here anyway, right?

A: Right now it's not being done, on the SUN. I'd say there will be some kind of table look up or you can use the equations.

Q: Are these the Martin tables?

A: No, this is the controller document. Somewhere in here it had the accuracies of the sensors.

{looking for a list in ICD of sensor accuracies and ranges}

Q: Are most the accuracies about 2% of full scale?

A: I thought we were supposed to be better than 1% of full scale.

{End of Tape}

Interview Session Date: 4/29/92
 Darrel Gaddy

[Discussing the post-test information in the package]

DG - This is a greenrun test. We were greenrunning the duct and the MFV actuator. We were doing a certification of the LOX pump, which is trying to get this [pump] design, [tested for] the flight timeline, so we could fly it. And then we have some valves and configurations of the facility that effects engine performance.

Fuel repress flow control valve on flight, we take part of the hydrogen flow that comes through the pump, goes through the MCC [cooling circuit] and goes to drive the low pressure fuel turbine, [and] we tap off here. And in flight we take this [the tapped off fuel] and stick it back ET [external tank] to pressurize the ET. We can have a flow rate here [pointing to the schematic the engine] of 1.3 lbs/second maximum or 0.2 lbs/second minimum. We were going to go for max. to min. at 300 seconds. And this effects the low pressure fuel turbine power and so it will effect the speed on the pump end, it will increase the energy to the pump. And so we will see that in the data.

And we will also do the same thing on the GOX repress flow. The GOX repress flow we get from the heat exchanger. We tap off a little bit after we go through LOX main impeller. We will go through the heat exchanger. We warm it, [so it becomes] it's gaseous and then we send it off to pressurize the LOX tank. But on the stand we just dump to overboard.

Question: Is there a valve upstream of this branch? On the facility side?

DG - There are no PIDs that tell you the valve position. No. We do have access to it, it called Facility D which is a vent [parameter] that will spit out a time in that this valve went from closed to open at this time, but we don't have a PID that goes from open to closed.

Also, the helium interface pressure we at 750 [psi] plus or minus 50 [psi] and obviously we are within that.

Hardwarewise we changed out the high pressure fuel duct which is running from the pump to the main fuel valves. We changed this duct out and [it is] likely to change performance a little bit. Also we changed some software and [this pre-test] tells you [what] change is [done].

We were looking here at HPOTP intermediate seal purge pressure which is a PID on the LOX pump, [it] was running a little bit low. It was low but the helium interface pressure which drives it was up so there was something in the pump that we were showing. That's a pre-start package.

Then we go into the start package where the data [is] from 0-6 seconds. We compare the current tests to the two previous tests with [the same] engine configuration, if we have it. If not we get the best comparison that we can and we choose the current and the two previous [test fires] for start and shutdown. This shows starts of PC going up and this is a phenomenon of fuel side oscillation. Fuel side oscillation effects the start. And we show the fuel turbine temps channel A to channel B. [We] show the fuel pump speed taking off here, [and] the lox turbine temps with its two channels, and the HPOP discharge pressure. We show the pressure since we don't have a speed and the pressure is [the] best indicator of speed.

Then a shutdown package here [is] trying to show the shutdown [process], but it pretty useless on a 1.5 second test, we can show it all during the start. Then we go through the anomalies that we have and instrumentation problems. That is usually what we present to the chief engineer.

DG - Another test we had was A1591. This test was a hydraulic lockup test. That is where we somehow remove the hydraulics from the engine and the valves are designed, so that as you remove the supply they will stay where they are. You can't command them open or closed. They lock up where they are. Then if you vary the inlet conditions, like the oxidizer inlet going into the low pressure pump you can't open up any of your control valves, FPOV or OPOV. So what you are going to see is main chamber pressure change for inlet pressures or the engine won't be able to keep PC with the inlet pressure changing.

(He is going through the test package)
This is an older test that shows allot more stuff [about] what the history of the components, we don't show that stuff anymore. This is what we were planning to do start up to 109% and then initiate hydraulic lockup here toward the end of the test. This is A1 gimble where the engine hangs and can spin it around like that (he is demonstrating a gimble procedure with his hands). We did two sets of gimbles. This is the software we put into the controller to get the right fuel flow or lox flow. This pre-start we used to show all the time, we don't have any need to show that anymore.

(Looking at the startup plots) [From these plots, it] looks like our start was fine. We compared it the two previous tests that we have on this engine and we can look at that [comparison] for tends. Or we have two sigma database, we like to see within the two sigma database. And this is a OPOV command and the OPOV command limit. You have your command, your command limit, and your actual two positions [for the OPOV]. The [OPOV] command at one place in your position tries to command [the valve to open], but your can't command above your limit. Your command limit is to keep you from [causing a] lox turbine [discharge temp] cutoff due to fuel leaks on flight.

Question: When you see something approaching that limit do you classify that as an anomaly.

DG - No.

We're running pretty close at this time [during start]. We don't think in flight, we don't want to lift off the pad with another margin there. [The controller] should bump it up out here once we [achieve the steady power level]. This [OPOV command limit] increases [changes] whenever you throttle.

Show fuel turbine temps versus the two sigma and LOX turbine temps versus the two sigma here.

(Now looking at shutdown plots) This one is the MCC PC, PID 63, went straight to zero after shutdown and these others [MCC PC traces] are real [hydraulically shutdown trace] measurements and I believe this one [the one that went straight down to zero] is an electrical lockup. You have different the controller of the different data, different places for different phases of operation, like hydraulic lockup. If there is truly hydraulic lockup you won't keep [maintain] PC, PID 63, but if its electrical lockup you'll substitute zero [value for PC at shutdown] at shutdown. Electrical lockup is where you still have control of the valves, but your not going to change the command, the command going to stay the same to the valve.

Question: Is this how you knew it went into electrical lockup?

DG - No, there is a report we get that the controller spits out, and it will say [at such a] time electrical lockup.

Question: Okay, is it that on the PID 287?

DG - No, I think it is on engine status word.

Response: I'm just trying to identify electrical lockup.

DG - It should be in the engine status work, [it] should tell you what phase of operation your in.

Question: So you weren't surprised to see this?

DG - The reason we do this is in actual flight, the orbiter looks at MCC PC to see if the engine is shutdown. And if you have electrical lockup, to get into electrical lockup you don't believe your MCC PC measurements or field flow meter measurement. You can't trust those, [so you] lock the engine up at the last place [sensor values]. [Since] you lock it [the engine] up, you can't trust them [the sensor measurements], so you don't know what the measurements are for PC. It could be way off or even high. Then if the orbiter comes back and tries to confirm shutdown, a failed sensor would tell it that the engine is still running when it's not. So we substitute a zero [value], so when the orbiter comes back to look at us we say we're shut down. That's why that [an electrical shutdown] substitutes zero [value for PC]. Then the hydraulic lockup we keep real MCC PC in here, because we believe the sensors. Any time you don't believe the sensors you set this [value] to zero. And that [is] the difference between electrical lockup [and hydraulic lockup]. Whenever you fail your pc sensor or fuel flow meter sensors, [it is an electrical lockup], and then the hydraulic lockup is when fail your FPOV and OPOV valves. Its really tricky to get it to go into electrical lockup or hydraulic lockup because whenever you remove hydraulics from the engine you lock everything up. The valves don't move and then the engine can be actually hydraulically lockup, but not know it until you have a 75 psi [difference] between PC reference and average of channel A measurement. When you get that 75 psi difference, it will send you into electrical lockup. Or, say the valve is locked-up and you start seeing PC dropoff. [You] try to open it and you can't open it and you're still commanding it open but it stays at the same position. Then if you have 6% difference on there [between the command value and valve position]. It is a race to see when you move hydraulics if PC 75 psi violates first or the valve position 6% violates. They have worked out little schemes that note how engine responded, so that they can send it into electrical lockup or hydraulic lockup.

Electrical lockup is the PC or fuel flow and you don't believe the measurements. And your looking for a 75 psi difference between PC reference and channel A average. The hydraulic lockup is when you don't think you can control your valve, your not sure where the position is. Your looking for, your command versus the actual position of 6% difference.

This is mainstage, we started at 100%, and we went up to 109%, and we throttled back down. Then you can see, this throttle difference here locked-up the engine and it started drifting.

Question: Now for this scheduled lockup you actually sent a command?

DG - No, they pulled a you actually fiddled with the pressure from hydraulic. They cut the hydraulic motor, that supplies 3000 psi hydraulics supply to the engine and [they] just shut it off.

Question: What are you testing here? How safe it is for the controller to order a hydraulic lockup?

DG - I think they were testing the design of the valve to make sure the valve would lock up and not drift, because they are not perfect. You lock them up and they leak a little bit and it allows them to close just a little bit and they drift while they are closing. We don't want them to drift. We're testing to see how much it would drift. So this one drifted a little bit and we lost some PC. Mixture ratios are already increasing.

This is something we do to the engine, we make the engine work harder by decreasing the pressure here. So it has to get the fuel flow up to this pressure and the more it has to pump [the pressure] up, the harder the engine has to work. So we vent this one down, to make it [the engine] work harder to pump the same pressure. And that's what we did there, we just vented the fuel tank down to make the engine work a little harder. And this is what we did on LOX

A3-243

side, we vented it down to make it work really hard, then we pressurized it to help it out some, and we came back to a nominal 80 psi.

These are the valve positions coming along here, the throttle here and showing this test versus another test. And they're reading different because its different engines, because each engine balances out little different.

This one [test fire], we were command limiting on the OPOV. That's why we saw a dip in [MCC] PC. Whenever we vented down this way, not far away from 200 seconds, we were making the engine work harder here and this is where starting repressurize. We had a little bit of margin, actually we didn't until we came down to this point. I think this is a start overshoot, comes to here. And this is where we starting venting and came back up that way. But anyway, we wouldn't open OPOV higher than this [the command limit], and so we couldn't send anymore power to the LOX pump. [So it's] not going to pump any harder, so it's going to drop off MCC PC. Then when we pressurize the inlet up, it [the pump] said "okay, I don't have to pump so hard, so I'm going to drift off a little bit, as long as I keep MCC PC here". This is called thrust limit whenever we bump this command versus command limit. Question: And that's what causes the deviations between the reference and actual chamber pressure?

DG - Yes. That's what causes this difference here. You can see whenever we're not on a [OPOV] command limit, [MCC] PC is in control but whenever we're thrust-limited, [MCC] PC drops off. So that's thrust limiting, when you go up to that [OPOV] command limit and [you] cannot open this one [valve] anymore, so you can decrease pressure there. Usually the reason why we thrust limit is we have fuel leaks. [During] fuel leaks, your flowing the same fuel through the fuel flow meter and it all has to come here. But if you have nozzle leaks that dumps the fuel out here [in the nozzle area], [so it] never makes it up here [MCC chamber], so you don't get as much thrust out of it. You get the most thrust if you dump it here [MCC chamber], but if you dump it down here will lose thrust or chamber pressure. We are controlling the chamber pressure, so if we're dumping fuel here [the nozzle area], it's not making it here [MCC chamber]. We have to make up [the lost chamber pressure] with LOX. That's why we thrust limit on the oxidizer preburner valve.

This is an old two sigma, we used to plot the two signal values during the main stage versus actual data.

Question: What trends would you see during hydraulic lockup, that you would not see before hydraulic lockup occurred?

DG - I'd see things like this. This is your CCV, which is the coolant control valve. It's command versus two positions. You can see the command staying the same but the position shifts.

Question: That is the bi-stability?

DG - No. That is that you're losing hydraulic pressure. On this one you have hydraulic forces that squeeze the valve wide open and you loose the hydraulic pressure and it's going to come off just a little bit.

Something on the hydraulic lockup, your valve position drifting away from command or command moving actual valve position, this is called valve drift.

Question: How would other then the main combustion chamber pressure lower where then any other engine phenomena that...?

DG - Mixture ratio changed. This is something that would happen [when] lockup the engine. The engine would see this pressure dropping off, [the engine would think it] needs more lox to make up pressure. So you start commanding the oxidizer valve open but it wouldn't move, so the command it open more.

Question: This is the command your looking at?

DG - Yes, it's the little squares and that's the command line. It says open me more, until it finally gets to about 6% [difference] and then says, "it's not moving, I'm not doing any good - lock me up".

Question: Do you actually see that PC dropping here?

DG - No. It's not in control because the valve is not moving, but you're trying to move it. Back here you can see the PC moving a little bit. The engine is trying to keep it up to this line, but it is falling off, so it's [the engine's] saying, "open me up, open me up".

Question: If an anomaly occurred other than what you expected in a crack or whatever, would hydraulic lockup would it be easy to see that type of thing happening?

DG - You mean if we have a crack here increase?

Response: Yeah.

DG - The crack is not associated with hydraulic lockup to the thrust movement. The way we look for cracks is if we're not sure if we have cracks that begin in start transient or shutdown transient, because you have the two transients you can crack here but you don't know if you have cracks from off performance, but you don't measure between the two with the engine operating. So if you have a crack here, the engine is going to try to make it up what LOX flow, and we can see the increase of LOX flow by increasing this power HPOP discharge pressure. And that how we look for leaks is by an increase in LOX flow. Usually takes about three pounds of LOX to make up a pound of fuel leaking out of here.

That's what we do whenever a gimble. [Pointing at the plot] We start in the middle and we just plot x versus y, and go out and rotate the engine. Randy usually makes it oblong this way.

These are some of the anomalies that we can see. Like this one...

Question: Anomalies unassociated with the fact your in hydraulic lockup?

DG - Yes. This is instrumentation anomaly where the [sensor] might be reading steady here, but this not real (he is pointing at a position on the plot), it looks like it come back in.

We have LOX injector pump pressure, we usually look at this pressure versus, LOX pump discharge pressure and we also have [MCC] PC. This one should be driving PC, so PC stays the same here there is no way its going to move, it should be driving PC. That's one of the logic [strategies] we look for in instrumentation problems, [to see if] anything else in the system is doing this.

Question: So you went and looked at PC and saw that is was flat,

DG - Yes. And we also looked at HPOP discharge pressure.

[end-of-tape]

A3-245

SSME Post-Test Diagnostic System Systems Section

**Final Report
Attachment #4**

**GENERIC Features Description and
TKCLIPS Users' Guide**

PTDS Feature Integration

T.W. Bickmore
Intelligent Software Associates, Inc.
3/27/95

I. Introduction

Most of the feature extraction routines have been modified so that they can be called as subroutines from any application. Routines which can only return zero or one feature were implemented as simple functions which return a flag indicating whether a feature was found or not. Routines which simply compute statistics were implemented as simple functions which just return the desired values. Routines which can return more than one feature on a given call do so via a callback function which is passed to them when they are invoked. As each feature is found, the callback function is called with a description of the feature. In this way, the application is free to do whatever it wants with information (e.g., update a database, or assert facts into an expert system shell).

In general, a generic feature function is called with the arrays of data and time samples and all parameters required by the feature, in addition to any descriptive information required by the callback function. Generic feature functions operate on one or more data arrays covering a single contiguous time interval, so if features need to be computed for several time intervals it is the responsibility of the caller to make separate calls for each such interval. Generic feature functions do not have any knowledge of how to access test data, nor do they have any knowledge of application-specific data structures. This is intended to make the features routines as adaptable as possible to future applications. All generic feature functions return FALSE if any problems were encountered, TRUE if all went well.

II. File Layout

GENERIC_features.c — Contains all generic features as described in the next section.

GENERIC_featurefits.c — Statistical routines used in the generic features, and made available for general use.

GENERIC_features.h — Header for above two files. These 3 should be stand-alone.

FEAT_features.c — The FEATURES executive and interfaces to the generic features.

CLIPS_features.c — The TKCLIPS interfaces to the generic functions.

III. Generic Feature Functions

In all cases integral values are 'int', floating point values are 'float'.

int GF_NoisyPid(time, sigmas, numpts, expert, testid, desc, thrust, pidstr,
gross_sigma, fine_sigma, callback)

Note: This is the FEATURES version.

Callback: void cf(expert, testid, start, end, desc, thrust, pidstr)

Feat callback fun: EHMS_WriteNoiseResults

int GF_IsFlat(time, data, sigmas, num_points, expert, testid, descrip, thrust, pidstr, callback)

Note: This is the FEATURES version.

Callback: void cf(expert, testid, descrip, start, end, slope, thrust, pidstr)

Feat callback fun: EHMS_WriteIsFlatResults

int GF_FindSpike(data, time, stddevs, numpts, bit_toggle, pid_range, percent_range,
rate, expert, testid, descrip, thrust, pidstr, callback)

Note: This is the FEATURES version.

Callback: void cf(expert, testid, descrip, start, end, mag, thrust, pidstr)

Feat callback fun: EHMS_WriteSpikeResults

int GF_FindErratic(data,time,stddevs,numpts,expert,expected_sigma,step_size,
 testid,descrip,thrust,pidstr,callback)

Note: This is the FEATURES version.

Note: The above must be called only for periods of linear LOX inlet pressure.

Callback: void cf(expert,testid,descrip,start,end,thrust,pidstr)

Feat callback fun: EHMS_WriteErraticResults

int GF_FindErratic2(data,time,stddevs,numpts,expert,sigma_threshold,
 testid,descrip,thrust,pidstr,callback)

Note: This is the TKCLIPS version.

Note: The above must be called only for periods of linear LOX inlet pressure.

Callback: void cf(expert,testid,descrip,start,end,thrust,pidstr)

int GF_FindPeak(data,time,numpts,min_peak,min_width,expert,testid,descrip,thrust,
 pidstr,peakht,taph,fwhm,offset,callback)

Note: This is the FEATURES version.

Callback: void cf(expert,testid,descrip,thrust,pidstr,peakht,taph,fwhm,offset)

Feat callback fun: EHMS_WritePeakRecord

int GF_FindConstantThrust(data,time,numpts,expert,testid,callback)

Note: The above must be called with 1-second averaged data for 287 from 5 to cut.

Note: This is the FEATURES version.

Callback: void cf(expert,testid,starttime,endtime,thrustlevel)

Feat callback fun: EHMS_FindConstantThrust_Callback

int GF_RedundChannelChk(data1,time1,numpts1,data2,time2,numpts2,
 threshold,index_per_time_seg,
 expert,testid,pid1,pid2,callback)

Note: Assumes signals have been start-aligned; data1 and data2 may have different rates.

Note: This is the FEATURES version.

Callback: void cf(expert,testid,pid1,pid2,start,end)

Feat callback fun: EHMS_WriteRedundResults

int GF_ZeroShiftCheck(time,data,numpts,low_limit,upper_limit,
 *average,*is_high,*is_low,*offset_high,*offset_low)

Note: This is the FEATURES version.

No callback function.

int GF_DifferentThan(time1,data1,sigmas1,time2,data2,sigmas2,num_points,
 num_comparison_sigmas,
 *is_different,*prob,*coeffs_within_bars,
 *differ_by_offset,*offset,*offset_sigma)

Note: The above must be called with data signals of the same rate which have already been aligned (num_points must be valid for both).

Note: This is the FEATURES version(?)

No callback used, since this either returns zero (is_different=FALSE) or one (is_different=TRUE) features.

int GF_Stats(data,time,numpts,*mean,*stddev,*min,*max)

Note: This is the TKCLIPS version.

No callback (doesn't make sense—always return just one set of values).

No FEATURES integration (don't know what to call with or where to put results).

int GF_PiecewiseLinear(data,time,numpts,testid1,pid1,testid2,pid2,step,
threshold,directional_only,expert,callback)
Callback: void cf(expert,testid1,pid1,testid2,pid2,starttime,endtime,startdata,enddata)
Note: This is the TKCLIPS version.
No FEATURES integration (don't know what to call with or where to put results).

int GF_FindLevelShift(data,time,stddevs,numpts,data_range,
testid,pid,expert,callback)
Callback: void cf(expert,testid,pid,starttime,endtime,
start_magnitude,shift_magnitude)
Note: This is the TKCLIPS version.
No FEATURES integration.

Note that the following FEATURES routines were not implemented as generic feature functions because they could not be resolved into analyses of single contiguous time segments (i.e., they appear to reason across multiple periods of LOX inlet pressure).

FindDrift – Try GF_PiecewiseLinear.

FindLevelShift — Try GF_FindLevelShift (old method).

IV. GF_featurefits

The following functions have been “genericized” from EHMS_featurefits. In almost all cases this simply involved changing the name of the function. This was done to make the modules GF_features and GF_featurefits stand-alone from the rest of the PTDS system.

GF_MakeFit	GF_GetNumberBasisFuncs
GF_GetMRQMemory	GF_FreeMRQMemory
GF_SetThreePointWindow	GF_GetFitInterval
GF_fit	GF_fitline (new)
GF_GetFitIntervalForSpikeCheck	GF_calculate_avg_sigma
GF_calc_stddev	

The result of this process is that the files FEAT_featurefits (FEATURES) and DATA_featurefits (TKCLIPS), and nrutil are now obsolete.

V. TKCLIPS User Functions

The following tkclips User Functions are now available:

V.1. TekBase Interface

(DB_connect) -- Connects to TekBase. Returns 0 if the connection could not be made, 1 if successful.

(DB_exec <TQLcommand>) -- Executes an arbitrary TQL command. The command CANNOT involve the transfer of data to or from the TQL server. Returns 1 if successful, 0 otherwise.

(DB_put <table> <fields> <values>) -- Adds a row of data to the specified table. Fields and values must each be a multi-field value. Currently, all fields must be specified for this to work properly. Returns 1 if successful, 0 otherwise.

(DB_get <table> <fields> <condition>) -- Retrieves rows of data into facts whose relation is the table name. Fields specifies the fields to be imported. Condition is a string specifying a valid TQL 'WHERE' clause. Returns 1 if successful, 0 otherwise.

(DB_disconnect) -- Disconnects from the current database, committing any actions taken during the session. This does not return any value.

V.2. Environment Access

(get_param [<n>]) -- If called with no argument returns the the 1st command line parameter used when CLIPS was invoked, otherwise NIL. If called with an integer argument, returns the Nth command line argument if there is one (where N=1 for the first argument), otherwise NIL.

(get_env <variable>) -- Returns the value of the specified variable in the Sun environment, or "".

(get_date) -- Returns the current date in a string.

(get_login) -- Returns the current user ID or "".

V.3. SSME DataFile Access and Feature Extraction

The following all attempt to access SSME data flatfiles directly and return their results as asserted facts. All return 1 if successful, 0 if any error occurred in processing. The environment variable 'NASA_TEST_DATA' must define a colon-separated list of paths to search for the datafiles (e.g., setenv NASA_TEST_DATA /data1:/data2:/hd1:/hd2). All input and output parameters are floats, except as specified below:

Type	Parameters
String	<descr>, <units>, <testid>, <pid>
Symbol	<label>
Integer	<windowSize>, <minPts>, <smoothWinSiz>, <stepSize>, <thrust_level>
"True" "False"	<withinErrorBars>, <DiffbyOffset>
"either_pid" "difference"	<checkType>
"upper" "lower"	<limitType>
0 (Full Sample)	<dataType>
1 (1sec Averaged)	
2+ (Smoothed, value=windowSize)	

Information about a sensor:*

(DATA_info <testid> <pid>)

-> (PIDINFO <testid> <pid> <start> <end> <descr> <units> <rate> <shutdown>)

Import raw data:*

(DATA_get <testid> <pid> <start> <end> <label>)

-> (DATA <label> <val> <sensor_reading>*)

-> (TIME <label> <val> <time_value>*)

Import averaged data:*

(DATA_average <testid> <pid> <start> <end> <label>)

-> (DATA <label> <val>...)

(TIME <label> <val>...)

(STDDEV <label> <val>...)

Import smoothed data:*

```
(DATA_smooth <testid> <pid> <start> <end> <>windowSize> <label>)
-> (DATA <label> <val>...)
    (TIME <label> <val>...)
```

Compute statistical summary:*

```
(DATA_stats <testid> <pid> <start> <end>) ;;Full sample only
-> (STATS <testid> <pid> <start> <end> <mean> <stddev> <min> <max>)
```

Compute statistical summary of the difference between two signals:*

```
(DATA_DeltaStats <testid> <pid1> <pid2> <start> <end>) ;;Full sample only
-> (DSTATS <testid> <pid1> <pid2> <start> <end> <mean> <stddev> <min> <max>)
```

Just fit a line through a signal:*

```
(DATA_FitLine <testid> <pid> <dataType> <start> <end>)
-> (LINE <testid> <pid> <start> <end> <offset> <slope>)
```

Determine thrust profile:*

```
(DATA_FindConstantThrust <testid>)
-> (F_THLEDE <testid> <start> <end> <thrust_level>)
```

Find erratic features:*

Note: Must only be called for periods of linear LOX inlet pressure.

```
(DATA_FindErratic <testid> <pid> <start> <end> <absStdDevThresh>) ;;Full sample only
-> (F_ERRAT <testid> <pid> <start> <end>)
```

Find level shift features:*

```
(DATA_FindLevelShift <testid> <pid> <dataType> <start> <end> <pidRange>)
-> (F_LEVSH <testid> <pid> <start> <end> <lastmag> <delta>)
```

Find peak features:*

```
(DATA_FindPeak <testid> <pid> <dataType> <start> <end> <minPeak> <minPts>)
-> (F_PEAK <testid> <pid> <time> <peak> <width(fwhm)> <offset>)
```

Find different than features:*

```
(DATA_FindDifferentThan <testid1> <pid1> <start1> <end1>
    <testid2> <pid2> <start2> <end2>
    <dataType> <numSigmasThresh>)
-> (F_DIFTHA <testid1> <pid1> <start1> <end1> <testid2> <pid2>
    <withinErrorBars> <DiffbyOffset> <offset> <offset_sigma>)
```

Compute piece-wise linear model:*

```
(DATA_FindPieceWise <testid> <pid> <dataType> <start> <end> <stepSize>
    <threshold> <directional_only>)
-> (SEGMENT <testid> <pid> <start> <end> <startval> <endval>) ;;Full sample only
;;If <directional_only> is 0, then any significant change in slope is noted.
;;Else, only changes through zero are noted.
;;Recommend: <step>=5, <dataType>=25, <thresh>=1/3range
```

Compute piece-wise linear model of the difference of two signals:*

```
(DATA_DeltaPieceWise <testid1> <pid1> <start1> <end1>
  <testid2> <pid2> <start2> <end2>
  <dataType> <stepSize> <threshold> <directional_only>)
-> (DSEGMENT <testid1> <pid1> <testid2> <pid2>
  <start> <end> <startval> <endval>)
;;If <directional_only> is 0, then any significant change in slope is noted.
;;Else, only changes through zero are noted.
```

Find spike features:*

```
(DATA_FindSpike <testid> <pid> <start> <end> <bitToggle> <Range> <perCent>)
;;Full sample only
-> (F_SPIKE <testid> <pid> <start> <end> <magnitude>)
;;<perCent> is portion of <Range> to use as a threshold.
```

SAIC method for Bistability:*

```
(DATA_DetectBistability <testid> <start> <end>) ;;Only call for PLs 65% or below.
-> (F_BISTAB <testid> <start> <end>)
```

New, improved method for Bistability:*

```
(DATA_DetectGreenBistability <testid> <start> <end> "209"|"210" "140"|"141")
-> (F_GREENBISTAB <testid> <start> <end>)
```

Check hard limits:*

```
(DATA_CheckUpperLimit <testid> <pid> <start> <end> <limit> <minPts>) ;;All full
sample only
(DATA_CheckLowerLimit <testid> <pid> <start> <end> <limit> <minPts>)
(DATA_CheckDifference <testid> <pid> <pid2> <start> <end>
  <limit> <minPts> <is_max>)
-> (F_RLVIOL <testid> <pid> {<pid2>|nil} <start> <end>
  <checkType> <limitType> <redline>)
;;<is_max> = 1 indicates that threshold is on maximum deviation, else minimum.
;;<minPts> is minimum number of consecutive data points which must exceed limit before
report.
```

Tell when a PID is within specified bounds:*

```
(DATA_FindInRange <testid> <pid> <start> <end> <low> <high> <minPts> <label>)
-> (F_INRANGE <testid> <pid> <start> <end> <label>)
```

Find IsFlat features:*

```
(DATA_IsFlat <testid> <pid> <dataType> <start> <end> <NumSigmasForFlat>)
-> (F_ISFLAT <testid> <pid> <start> <end> <slope>)
Note: EHMS_NumSigmasForFlat is 3.0
```

Find DeltaLevelShift features:*

```
(DATA_DeltaLevelShift <testid1> <pid1> <start1> <end1> <testid2> <pid2> <start2>
  <end2> <dataType>)
-> (F_LEVSH <testid> <pid1-pid2> <start> <end> <lastmag> <delta>)
;;Recommend: <dataType>=(int)rate if pids have same rate.
```

Find DeltaDifferentThan features:

```
(DATA_DeltaDifferentThan <testid1> <pid1a> <pid1b> <start1> <end1>
                        <testid2> <pid2a> <pid2b> <start2> <end2>
                        <dataType> <numSigmasThresh>)
-> (F_DIFTHA <testid1> <pid> <start1> <end1> <testid2> <compPid2>
    <withinErrorBars> <DiffbyOffset> <offset> <offset_sigma>)
```

Fit a line to the difference between two signals:*

```
(DATA_DeltaLine <testid1> <pid1> <start1> <end1> <testid2> <pid2> <start2> <end2>)
-> (DLINE <testid1> <pid1> <start1> <testid2> <pid2> <start2> <offset> <slope>)
```

Find Noise features:*

```
(DATA_FindNoise <testid> <pid> <start> <end> <sigmaThresh>) ;;1s Avg only
-> (F_NOISE <testid> <pid> <start> <end>)
```

Zero shift check:*

```
(DATA_ZeroShiftCheck <testid> <pid> <start> <end> <lower> <upper>)
-> (F_ZEROSC <testid> <pid> <start> <end> <offset>)
```

Redundant channel check:

```
(DATA_RedundantChannelCheck <testid> <pid1> <pid2> <start> <end>
                        <threshold> <index_per_time_seg>)
-> (F_RD_CC <testid> <pid1> <pid2> <start> <end>)
```

VI. Unix Command-Line Utilities

In addition to testinfo, listpids, getpid, plot, plotdiff, and twoplots, the following command-line utilities have been implemented to test out the generic features:

stats <testid> <pid> <start> <end> — Prints out statistics for the signal described.

deltatstats <testid> <pid1> <pid2> <start> <end> — Prints out statistics for the difference between the two described signals (1 - 2).

thrust <testid> — Prints out the thrust profile for the indicated test.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE August 1995	3. REPORT TYPE AND DATES COVERED Final Contractor Report	
4. TITLE AND SUBTITLE SSME Post Test Diagnostic System Systems Section			5. FUNDING NUMBERS WU-584-03-11 C-NAS3-25883	
6. AUTHOR(S) Timothy Bickmore				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Aerojet Propulsion Systems P.O. Box 13222 Sacramento, California 95813-6000			8. PERFORMING ORGANIZATION REPORT NUMBER E-9829	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191			10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA CR-198375	
11. SUPPLEMENTARY NOTES Project Manager, June F. Zakrajsek, Space Propulsion Technology Division, NASA Lewis Research Center, organization code 5310, (216) 433-7470.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Categories 15, 16, and 20 This publication is available from the NASA Center for Aerospace Information, (301) 621-0390.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) An assessment of engine and component health is routinely made after each test firing or flight firing of a Space Shuttle Main Engine (SSME). Currently, this health assessment is done by teams of engineers who manually review sensor data, performance data, and engine and component operating histories. Based on review of information from these various sources, an evaluation is made as to the health of each component of the SSME and the preparedness of the engine for another test or flight. The objective of this project—the SSME Post-Test Diagnostic System (PTDS)—is to develop a computer program which automates the analysis of test data from the SSME in order to detect and diagnose anomalies. This report primarily covers work on the Systems Section of the PTDS, which automates the analyses performed by the systems/performance group at the Propulsion Branch of NASA Marshall Space Flight Center (MSFC). This group is responsible for assessing the overall health and performance of the engine, and detecting and diagnosing anomalies which involve multiple components (other groups are responsible for analyzing the behavior of specific components). The PTDS utilizes several advanced software technologies to perform its analyses. Raw test data is analyzed using signal processing routines which detect features in the data, such as spikes, shifts, peaks, and drifts. Component analyses are performed by expert systems, which use "rules-of-thumb" obtained from interviews with the MSFC data analysts to detect and diagnose anomalies. The systems analysis is performed using case-based reasoning. Results of all analyses are stored in a relational database and displayed via an X-window-based graphical user interface which provides ranked lists of anomalies and observations by engine component, along with supporting data plots for each.				
14. SUBJECT TERMS Space shuttle main engine; Expert systems; Data reduction; Systems analysis			15. NUMBER OF PAGES 420	
			16. PRICE CODE A18	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT	